NAVAL
POSTGRADUATE
SCHOOL

MONTEREY, CALIFORNIA

SYSTEMS ENGINEERING CAPSTONE PROJECT REPORT

PREVENTING PIRATES FROM BOARDING COMMERCIAL VESSELS – A SYSTEMS APPROACH

by

Team Pirates
Cohort 311-111A4 and 311–131A

September 2014

Project Advisor:	Paul Shebalin
Second Reader:	Richard Millar

REPORT DOCUMENTATION PAGE		*Form Approved OMB No. 0704–0188*

1. AGENCY USE ONLY *(Leave blank)*	**2. REPORT DATE** September 2014	**3. REPORT TYPE AND DATES COVERED** Capstone Project Report
4. TITLE AND SUBTITLE PREVENTING PIRATES FROM BOARDING COMMERCIAL VESSELS – A SYSTEMS APPROACH		**5. FUNDING NUMBERS**
6. AUTHOR(S) SE Cohort 311-111A4 and 311–131A, Team Pirates		
7. PERFORMING ORGANIZATION NAME(S) AND ADDRESS(ES) Naval Postgraduate School Monterey, CA 93943-5000		**8. PERFORMING ORGANIZATION REPORT NUMBER** NPS-SE-XZY
9. SPONSORING /MONITORING AGENCY NAME(S) AND ADDRESS(ES) N/A		**10. SPONSORING/MONITORING AGENCY REPORT NUMBER**

11. SUPPLEMENTARY NOTES the views expressed in this thesis are those of the author and do not reflect the official policy or position of the Department of Defense or the U.S. Government. IRB protocol number ____N/A____.

12a. DISTRIBUTION / AVAILABILITY STATEMENT Approved for public release; distribution is unlimited	**12b. DISTRIBUTION CODE**

13. ABSTRACT (maximum 200 words)

Piracy represents a serious threat to modern maritime traffic, causing significant financial losses as well as loss of life. The system's proposed area of operation is the waters of Indonesia, as current antipiracy solutions are not feasible due to the region's unique physical geography. Worldwide deployment is possible with minimal modifications. The systems engineering process was used to identify a system that effectively and economically prevents pirates from boarding commercial vessels. A model of the operational environment was developed in MATLAB to run simulations designed to estimate the relative effectiveness of each assessed countermeasure. A cost analysis was performed on the most effective system configurations to determine economic feasibility; the best-value system was recommended. The results of the project indicated that the P-Trap countermeasure, designed to entangle the pirate's propellers with thin lines, is both effective and economically viable for wide-scale deployment. The further addition of a fire hose system using net projectiles to increase the difficulty of boarders to climb onto the vessel was found to enhance the system effectiveness, while remaining cost-effective.

14. SUBJECT TERMS modeling and simulation, system integration, system architecture, Southeast Asia, Indonesia, piracy, boarding, countermeasures, hijacking, trade study	**15. NUMBER OF PAGES** 147
	16. PRICE CODE

17. SECURITY CLASSIFICATION OF REPORT Unclassified	**18. SECURITY CLASSIFICATION OF THIS PAGE** Unclassified	**19. SECURITY CLASSIFICATION OF ABSTRACT** Unclassified	**20. LIMITATION OF ABSTRACT** UU

NSN 7540–01-280-5500 Standard Form 298 (Rev. 2–89) Prescribed by ANSI Std. 239–18

THIS PAGE INTENTIONALLY LEFT BLANK

PREVENTING PIRATES FROM BOARDING COMMERCIAL VESSELS – A SYSTEMS APPROACH

Cohort 311-111A4 and 311–131A/Team Pirates

Juan Cabungcal	David Kaniss	Chris Laing
Keith Mastran	Jason Powell	Nathaniel Quijano
Eric Rosenberg	Greg Walsh	

Submitted in partial fulfillment of the
requirements for the degree of

MASTER OF SCIENCE IN SYSTEMS ENGINEERING

from the

NAVAL POSTGRADUATE SCHOOL
September 2014

Lead editor:	Eric Rosenberg
Assistant editor:	David Kaniss

Reviewed by: Dr. Paul Shebalin, Project Advisor — Dr. Richard Millar, Second Reader

Accepted by:
Dr. Cliff Whitcomb
Chair, Systems Engineering Department

THIS PAGE INTENTIONALLY LEFT BLANK

ABSTRACT

Piracy represents a serious threat to modern maritime traffic, causing significant financial losses as well as loss of life. The system's proposed area of operation is the waters of Indonesia, as current antipiracy solutions are not feasible due to the region's unique physical geography. Worldwide deployment is possible with minimal modifications. The systems engineering process was used to identify a system that effectively and economically prevents pirates from boarding commercial vessels. A model of the operational environment was developed in MATLAB to run simulations designed to estimate the relative effectiveness of each assessed countermeasure. A cost analysis was performed on the most effective system configurations to determine economic feasibility; the best-value system was recommended. The results of the project indicated that the P-Trap countermeasure, designed to entangle the pirate's propellers with thin lines, is both effective and economically viable for wide-scale deployment. The further addition of a fire hose system using net projectiles to increase the difficulty of boarders to climb onto the vessel was found to enhance the system effectiveness, while remaining cost-effective.

THIS PAGE INTENTIONALLY LEFT BLANK

TABLE OF CONTENTS

LIST OF FIGURES

THIS PAGE INTENTIONALLY LEFT BLANK

LIST OF TABLES

THIS PAGE INTENTIONALLY LEFT BLANK

LIST OF ACRONYMS AND ABBREVIATIONS

BPS	Boarding Prevention System
C2	Command and Control
CONOPS	Concept of Operations
COTS	Commercial off-the-Shelf
DOE	Design of Experiments
DOD	Department of Defense
FY	Fiscal Year
ICC	International Chamber of Commerce
IMB	International Maritime Bureau
KPP	Key Performance Parameter
LCC	Life-Cycle Cost
MOE	Measure of Effectiveness
MOP	Measure of Performance
M&S	Modeling and Simulation
RPG	Rocket Propelled Grenade
SA	Situational Awareness
SE	Systems Engineering
SoS	System-of-Systems

THIS PAGE INTENTIONALLY LEFT BLANK

EXECUTIVE SUMMARY

Maritime piracy is often thought of as a historical problem, but it is a significant contemporary threat to international commerce. Pirates are able to capture cargo ships and tankers worth millions of dollars and ransom the ship and its crew back to the home nation. While naval deployments have successfully controlled hotspots of pirate activity such as Somalia, this stratagem is not universally effective. Pirates in the seas around Indonesia are able to use local geography to reduce the effectiveness of naval task forces. In part due to this, piracy in Indonesian waters has surged to nearly half of all reported pirate attacks in recent years. This report focuses on the identification of a solution that will prevent pirates from boarding commercial shipping vessels.

This project involves a wide range of stakeholders interested in some aspect of the system. Merchant crews, shipping companies, and international maritime organizations were identified as key stakeholders. Inputs were obtained from the key stakeholders, and were used to generate system requirements. The analysis of top-level user requirements showed that the system could be considered successful in deterring piracy if pirate boarding is prevented for the period of time required for aid to arrive. Pirate deterrence can be accomplished by the second level requirements of impeding pirate entry routes, forcing pirates away from the target merchant vessel, or degrading pirate capabilities.

Research of the current and theorized pirate countermeasures used to deter or prevent boarding resulted in a comprehensive database of twenty-five (25) countermeasures. The list of countermeasures was reduced by rating each item on Measures of Performance (MoP) and Measures of Suitability (MoS) such as Time to Deploy, Ease of Use, Maintenance, Cost, and Logistics. The results indicated that the five countermeasures should be modeled to determine effectiveness: Razor Wire, P-Traps, Water Curtains, Fire-hoses, and Compressed Air Cannons.

A model of the operational environment was coded using MATLAB that utilizes a predator-prey relationship to represent the pirate vessels and the commercial ship. The model assumed that multiple hostile vessels would engage in an attack, with the intent to overrun the target's defenses. For each countermeasure configuration a functional flow of events versus a simulated pirate attack was performed using MATLAB software. System configurations were developed to estimate the relative effectiveness of each configuration, as well as the cumulative effects of employing multiple countermeasures simultaneously.

Twenty-four system configurations were selected out of thirty-two possible by filtering out the eight system configurations in which neither of the passive defense countermeasures was used; utilizing only complex, active countermeasures would unnecessarily increase both manpower requirements and cost, and would be rejected as gold-plating. Each system configuration was modeled using MATLAB and a simulation of a pirate attack on a commercial vessel was run 1,000 times for each modeled system configuration to determine the system configuration most effective at preventing pirate capture of the commercial vessel. A cost analysis of each system configuration was also performed, and used to determine the overall desirability of the system configuration.

The results of the simulations and cost analyses showed three configurations that maximized cost-effectiveness. Usage of the P-Trap countermeasure combined with the Compressed Air Cannon provided a success rate of 97.3% with a five-year cost of \$1.164M/ship. A slightly more effective system configuration consists of the P-Trap countermeasure combined with the Fire Hose, with a success rate of 99.4% and a five year cost of \$1.341M/ship. Adding the Anti-Piracy Curtain to the P-Trap and Fire Hose countermeasures improves the success rate to 99.7%, but increased the system cost to a five-year cost of \$1.576M/ship.

ACKNOWLEDGMENTS

The Pirates team wishes to express our gratitude to our capstone advisor, Doctor Paul Shebalin, for his guidance, suggestions, and feedback throughout this project. This project could not have been completed without his generous and professional assistance. We would similarly like to thank Doctor Richard Millar, who reviewed our paper and provided valuable feedback.

The team would also like to thank Heather Hahn, our distance learning coordinator, for her clear explanations of the capstone process and her eagerness to assist the team with any questions or concerns.

Finally, the team would also like to thank Professor Ron Carlson for his talent in explaining the concepts, philosophy, and methods of Systems Engineering during his Introduction to Systems Engineering course. This class laid the groundwork for the remainder of the program, and ensured that students enrolled in the Naval Postgraduate School gained an understanding of systems engineering tools and processes.

THIS PAGE INTENTIONALLY LEFT BLANK

I. INTRODUCTION

A. BACKGROUND

Modern-day piracy presents a difficult and expensive problem for international trade, driving up shipping costs and placing merchant crews in peril. According to Peter Chalk in his analysis of maritime security for the RAND Corporation, pirates board and capture maritime traffic ranging from small luxury yachts to fully loaded supercargo ships carrying multi-million dollar cargoes (Chalk 2008). High-profile incidents where cargo, ship, and crew are held for ransom have made international news and have been dramatized in film, such as the 2013 action-thriller movie, *Captain Philips*. Far more numerous are incidents of mere thievery, where valuables and cargo are stolen under threat of force or stealthily, under the cover of darkness. The financial costs incurred from these criminal acts significantly increase shipping costs due to higher insurance premiums, increased transportation costs due to longer trade routes bypassing piracy prone areas, and loss of operation of attacked ships. As noted in *Hellenic Shipping News*, the economic effects of piracy cost the worldwide economy an estimated $6B in 2012 (*Hellenic* 2014).

The International Maritime Bureau (IMB), a department of the International Chamber of Commerce (ICC), has published an annual report for the last two decades detailing maritime piracy. According to the report detailing 2013 incidents, Southeast Asia accounts for almost half of the total worldwide pirate attacks, while Africa accounts for 30% of the total incidents. The results from these studies (Table 1) show that Indonesia is an obvious area to target for an anti-piracy system or program, as over 48% of piracy-related incidents occurred in Indonesian sovereign waters and since Southeast Asia is the sole area where pirate attacks are on the increase (IMB 2013).

Table 1. Regions of Actual and Attempted Pirate Attacks Worldwide 2009–2013

Adapted from ICC IMB 2013 annual report, this table shows reported pirate attacks in each region for the last five years, as well as each region's percentage of global pirate attacks in 2013.

	2009	**2010**	**2011**	**2012**	**2013**	**% of total Attacks in 2013**
SE Asia	46	70	80	104	128	48.5%
Africa	266	259	293	150	79	29.9%
India	30	28	16	19	26	9.8%
South America	37	40	25	17	18	6.8%
Far East	23	44	23	7	13	4.9%
Rest of World	8	4	2	0	0	0.0%
Total Attacks	**410**	**445**	**439**	**297**	**264**	

To date, the most successful method for combatting the piracy problem has been naval patrols. The waters around the Horn of Africa were a hotspot for piracy in the early 2000s, resulting in the United Nations Security Council assembling a twenty-five nation task force, known as Combined Task Force 151 (CTF-151), to combat the pirate threat. CTF-151 succeeded in reducing the number of piracy incidents in the affected region due to the Gulf of Aden's geography, consisting of a long and narrow strait used by commercial traffic. The strait contained limited options for a pirate to evade a naval vessel and few areas suitable for a pirate headquarters. Unfortunately, this strategy is unlikely to prove effective in the burgeoning piracy hotspot of the Indonesia region.

The deployment of naval forces in the Indonesia region would face difficulties beyond those seen in Somalia, as the physical geography significantly differs from the Gulf of Aden. Andrew Manners, a piracy analyst for the region, noted that the area to be patrolled is significantly larger, shipping traffic is much less concentrated, and thousands of islands and coastal mangrove swamps with relatively small waterways provide ample opportunities for pirates to evade capture (Manners 2014). As such, the number of naval vessels required to effectively patrol the region would be a massive and expensive undertaking. In addition, a second large force of patrol boats would be required to pursue the pirates through shallow, swampy channels. Finally, the political and diplomatic considerations required for a large naval deployment in Indonesian waters are likely to be significant due the number of competing sovereign states. Therefore, a non-taskforce based solution is required to effectively combat piracy in this region.

1. Piracy in Indonesian Waters

Four conditions are required for piracy to thrive: a target-rich environment, a lack of strong government, weak or corrupt local law enforcement, and a high reward-to-risk ratio (Samatar 2014). Several areas near high-traffic shipping lanes, such as Southeast Asia and West Africa, fit the stated conditions and have become hotbeds for pirate activity (Ben-Ari 2013). Indonesia is a prime example of an environment perfectly suited to incubate piracy. As noted by Eric Frécon in his fieldwork amongst the communities of the Riau islands near Singapore, the local government is corrupt, and the local security forces lack the resources to deter piracy. Specifically, he notes that in the coastal community of Kampung Hitam, "the police only have at their disposal small sampans with only one outboard motor, when, in comparison, pirates often have two or three. often shabbily dressed without proper uniforms, the policemen spend their time idling…rather than clamping down upon pirate activities." (Ong-Webb 2006, 73). While improvements could be made to strengthen local forces, such efforts are not likely to yield results swiftly or cheaply.

In order to better characterize the vessels involved in a typical pirate attack, a gross tonnage (GT) analysis was performed for both local maritime traffic and commercial vessels to determine typical scenario data. According to a 2012 report published by Equasis, commercial shipping vessels are categorized into four classes: small (under 500 GT), medium (between 500 GT to 25,000 GT), large (25,000 GT and 60,000 GT), and very large (over 60,000 GT). Forty-six percent of global merchant vessels are classed as medium, while small vessels take a close second place with 36%. Large and very large vessels are considered less applicable for this study, comprising only 12% and 6%, respectively, of the total worldwide number of merchant vessels (Equasis 2012). Indonesian fishing vessels are significantly smaller than commercial merchant ships: out of roughly 300,000 registered fishing vessels, 40% are small (under 5 GT), 24% fall between 5 and 10 GT, 18% fall between 10 and 30 GT, while 18% are above 30 GT (Lymer 2009).

The results of the GT analysis highlighted the disparate sizes of local vessels versus the commercial vessels, and were used to determine a baseline pirate attack scenario. Small and medium commercial shipping vessels represent 83% of the worldwide fleet. Because medium ships contain more enemy approach routes and thus a larger area to protect, they were used to define the typical targeted ship. The typical pirate vessel is more difficult to quantify due to the wide spread of GT, so a smaller size (5 GT) was chosen for inclusion in scenario design to maximize the boarding threat as smaller vessels are faster and more likely to be used in swarming attacks.

Indonesian pirates use similar methods as those in other regions of the world, consisting primarily of pretending to be a fishing vessel until the target ship is close enough for an attempt to board (Allen 2013). Indonesia is currently the second largest producer of seafood worldwide (FAO 2014), so the pirates are able to easily blend in amongst genuine fishing vessels to avoid anti-piracy enforcement or alarming potential targets. Since most pirate vessels are repurposed fishing vessels, it is virtually impossible to obtain a positive identification of the pirate before an attack commences. A pirate can track, survey, and approach their targets with impunity, only revealing hostile intentions once close to a target. While the number of pirates in an attack can vary considerably, the weaponry used typically consists of small arms, knives, and grenades. Pirate attacks have ranged from a small number of people in a single boat to larger groups with multiple craft in more organized attacks (Lamb 2011).

Pirates in this region are often successful once they begin an attack. Figure 1 depicts high-traffic straits and all reported pirate attacks in Indonesian waters during 2013 (ICC IMB 2013). Yellow pins represent attempted attacks, orange pins show a boarded ship, and red pins depict a hijacked ship. However, while the figure shows pirate attacks, it does not show how many ships traveled through this region safely and what the relative danger level is to estimate the probability of attack by pirates, a traffic analysis conducted by the U.S. Energy Information Administration of four large straits was analyzed and recorded in Table 2: the Strait of Malaccca, the Sunda Strait, and the combined Makassar and Lombok Straits (EIA 2012). The ships attacked in 2013 were divided by the total number of vessels to obtain the percentage of ships attacked. Additionally, the financial impact was estimated by determining the average worth of the cargo and multiplying it by the number of attacks to obtain the total dollar amount that could be seized by pirates. While the Straits of Malacca contained the most traffic and thus the most attacks, the combined Makassar and Lombok Straits had significantly higher attack rates and ship cargo worth. The results of the analysis indicate due to the low piracy prevention rate, a potential need for a new method of prevention exists in this region. However, the relatively low chance of pirate attacks shows that any solution must be cost-effective to appeal to a profit-driven industry.

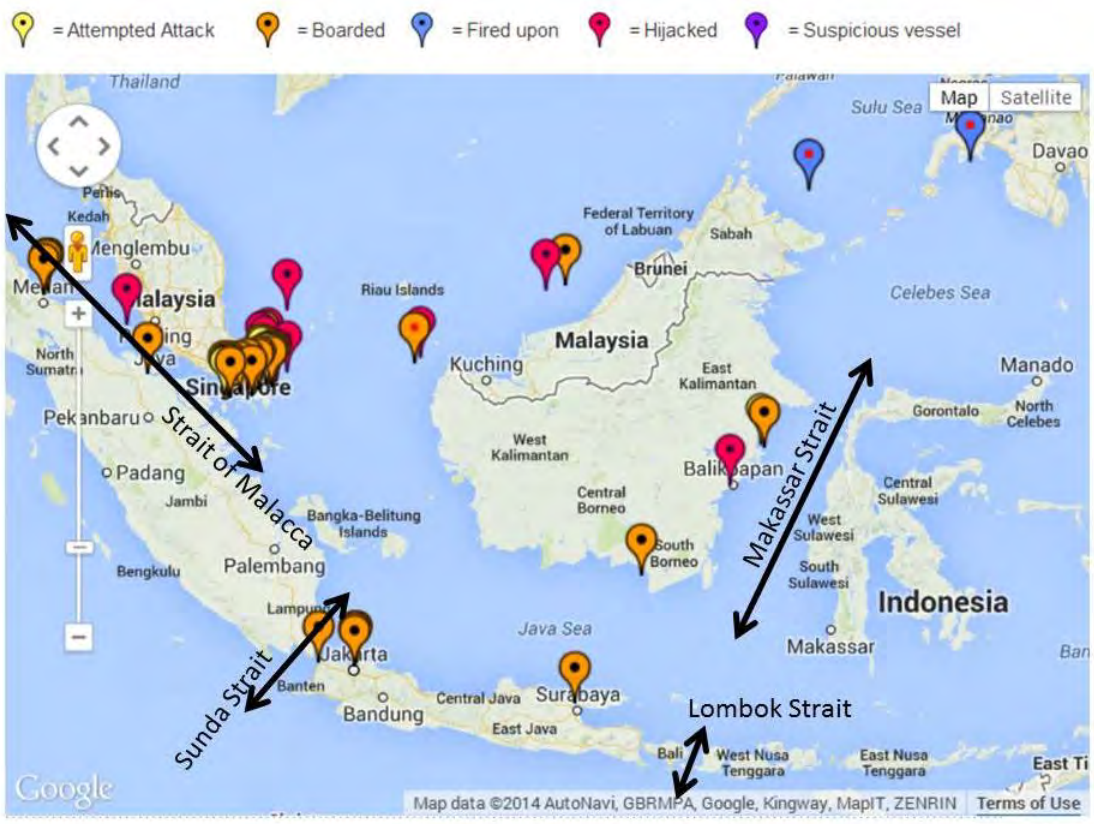


Figure 1. 2013 Pirate Attacks in Indonesia

Adapted from ICC Piracy Reporting Center 2013, this figure depicts pirate attacks in Southeast Asia during 2013.

Table 2. Analysis of Traffic, Cargo, and Danger in Indonesian Straits

Adapted from Nanyang Technological University, the U.S. Energy Information Administration, and the Commander of U.S. Pacific Fleet, this table shows the percent of ships attacked, as well as the average ship value and potential financial loss of the cargo.

	Yearly Vessels	Yearly cargo worth ($B)	2013 Attacks	% Ships Attacked	Average Ship Value ($M)	2013 Affected ($M)
Strait of Malacca	60000	1300	55	0.09%	$21.67	$1,191.67
Makassar and Lombok Straits	420	40	7	1.67%	$95.24	$666.67
Sunda Strait	2280	5	8	0.35%	$2.19	$17.54

The area around Singapore, shown in Figure 2, is the largest regional piracy hotspot, with the bulk of pirate attacks occurring on either side of the heavily travelled Singapore Strait. On the western outlet, a typical pirate attack involves a single wooden boat with no more than five pirates armed with long knives attacking a slow moving commercial vessel transiting out of the strait. In all of these attacks, the pirates stole mechanical parts and crew valuables but always fled after being spotted by the target ship's crew.

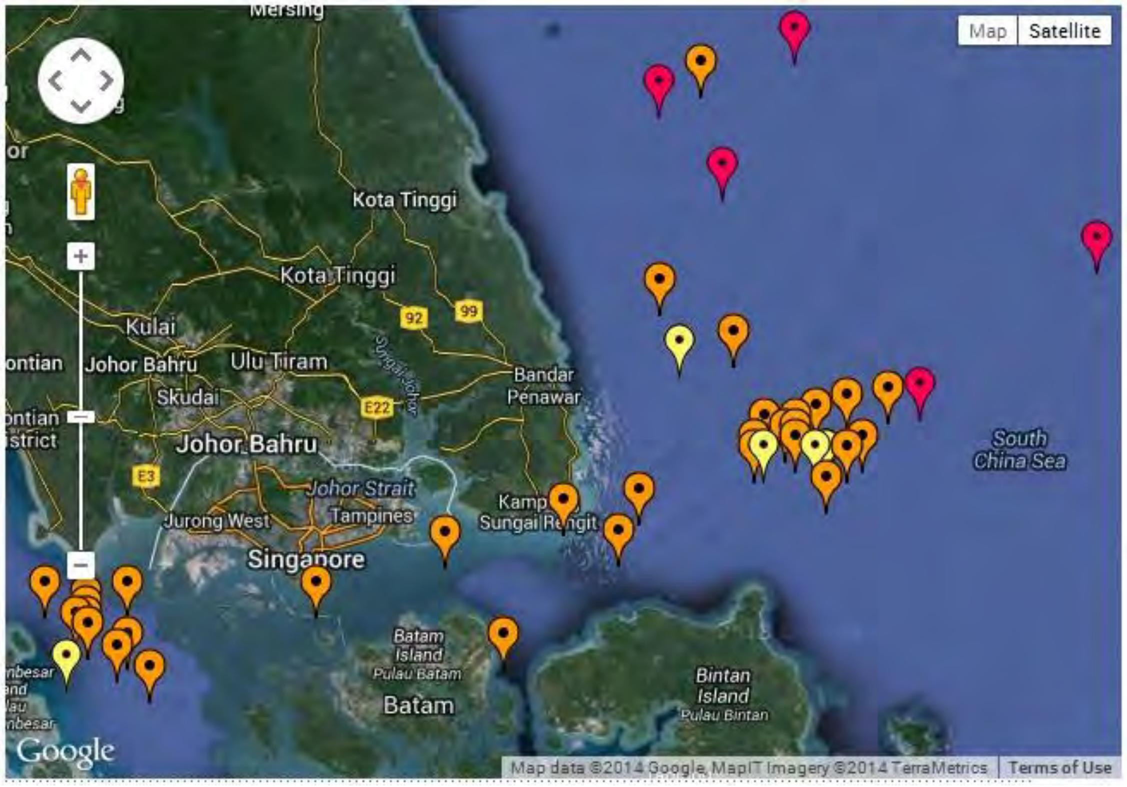


Figure 2. 2013 Pirate Attacks near Singapore

Adapted from ICC Piracy Reporting Center, this figure depicts reported pirate attacks near Singapore in 2013. Many attacks near western Singapore waters consist of snatch-and-grabs by a small number of pirates near a port. Attacks in eastern waters involve full speed chases and hijacking.

The eastern outlet of the Singapore Strait and the shipping lanes feeding into it from the South China Sea are home to pirates with considerably better equipment, manpower, and coordination. A typical pirate attack involves multiple pirate craft with larger engines, capable of chasing down a commercial vessel travelling at full speed (21-

25 knots) in open waters. The pirates manning these craft are more numerous and possess small arms as well as long knives. These pirates targeted isolated commercial vessels (at least an hour from any assistance) at twenty to fifty nautical miles from shore. Upon successfully boarding, merchant crews were always subdued and bound, the communication system of the target vessel was destroyed, and cargo transferred off to a pirate cargo vessel (typically a bunker ship). Usually the merchant crew and ship were released after the theft was completed, with the exception of a single incident where a tug was stolen and the crew set adrift on a barge. Because these pirates caused significantly more economic damage, further research was focused on the open waters pirates to discern the typical types of pirate attack in the region

2. Pirate Attack Strategies

According to Elleman, Forbes, and Rosenberg, pirate attacks near Indonesia employ a similar attack strategy. Each attack is typically conducted on a moving commercial ship, using the cover of darkness to avoid detection. Attacks typically fall between the hours of 0100 and 0600 to ensure that the majority of the crew is asleep. When the pirates are within range of the vessel, grappling hooks are used to board the target ship; the boarding party makes their way on to the vessel, and then subdues the lookouts. Once the pirates have boarded, they quickly make their way to the crew quarters to subdue the crew, pilfering any valuables encountered. Once the crew is neutralized, the pirates will employ one of three different strategies: cargo theft, kidnapping for the purpose of ransom, or seizure of the ship (Elleman, Forbes, and Rosenberg 2010).

This first strategy, cargo theft, is the most common of the three. The pirates transfer part or all the of the ship's cargo onto their own vessel(s), requiring from three to seven hours. Once the pirate ship is full of stolen goods, the pirates will release the crew and depart.

The second possible strategy consists of attempting to extort a ransom from the crew's employer by taking the crew hostage. Key crew members, such as the captain or pilot, are taken as hostages and removed from the ship to a remote, land-based location. The pirates then begin negotiations with the hostages' employer for the release of the crew members. In most cases the hostages are released once the ransom amount is received. A variation on this approach is for the pirates to remain on the ship and ransom the ship, crew, and goods back to the shipping company for an exorbitant sum.

The third possible action is a complete high-jacking and confiscation of the ship, crew and cargo. Once the ship is under the pirates control, it is taken to a hidden location where the cargo is removed, the crew is either killed off or ransomed, and the ship is repainted by the pirates for use in future attacks. The ship's navigation system is modified to squawk false identifications, allowing the pirates to trick other vessels into allowing them to get close enough to mount an attack. This method is virtually unheard of in Indonesian waters due to the additional time requirements and risk for the pirates.

B. PROBLEM STATEMENT

Indonesian geography presents unique challenges that prevent traditional anti-piracy methods from being deployed effectively. While current anti-piracy approaches, such as CTF-151, have reduced pirate attacks in traditional hotspots such as Somalia, detecting and responding to pirate attacks in time to prevent or defeat the attack is problematic for counter-piracy military and police forces. Commercial vessels in Indonesian waters often must fend for themselves against prepared and determined foes. Because of this, commercial vessels in Indonesian waters need an onboard means of preventing or delaying pirates from boarding.

C. RESEARCH QUESTIONS

The central research question is: What systems engineering solution will prevent or reduce the success of maritime piracy in Indonesian waters? the following research sub-questions were considered for this project:

- What capability gaps need to be addressed in current anti-piracy approaches?
- What characteristics, signatures and patterns mark a pirate vessel?
- What tactics, equipment, and methods are used by pirates during an attack?
- What distinguishes piracy in Indonesia from that in other regions?

D. PROJECT OBJECTIVE

The purpose of this project was to define and simulate a Boarding Prevention System (BPS) for use on commercial vessels affected by piracy in Indonesian waters. The results of the project simulation and cost modeling were used to recommend a specific system configuration of countermeasures to be employed by the BPS.

E. SCOPE

The definition and simulation of the BPS focused on determining the efficacy of multiple countermeasures working in concert. The capabilities, costs, reliability, maintainability, and supportability of the countermeasures were the main focus of investigation. Effects from crew training and capability were not examined and are a potential area for future study.

In order to develop the BPS simulation, all the noted research questions were investigated. Analysis of the piracy problem in the Indonesian region was limited to pirate vessels, equipment, tactics, targeted commercial ships, and how the unique geography of the region affects anti-piracy efforts. Areas for further study could include the effectiveness of the local navies (Malaysian, Indonesian, Singaporean) working in concert to combat regional piracy. Additionally, socio-economic solutions to the regional piracy problem could be investigated.

II. APPROACH

In meeting the research objective stated above, this project identified an unmet commercial need, and applied a modified Systems Engineering methodology to refine, analyze, and address this need. The approach used for this project is detailed in this chapter and the results are shown in Chapter III.

A. SYSTEMS ENGINEERING PROCESS OVERVIEW

A tailored System Engineering (SE) process, summarized in Figure 3, was used to investigate and combat maritime piracy in troubled regions throughout the world, with a focus on commercial vessels in Indonesian waters. Six sequential main phases were identified, beginning with needs analysis and concluding with a recommended solution in the final report; each phase contained multiple sub-phases that were executed simultaneously. A tailored process model was chosen due to the limited scope of the project, and was loosely based on a Systems Engineering "Vee" methodology created by Forsberg and Mooz (Blanchard and Fabrycky, 2011.)

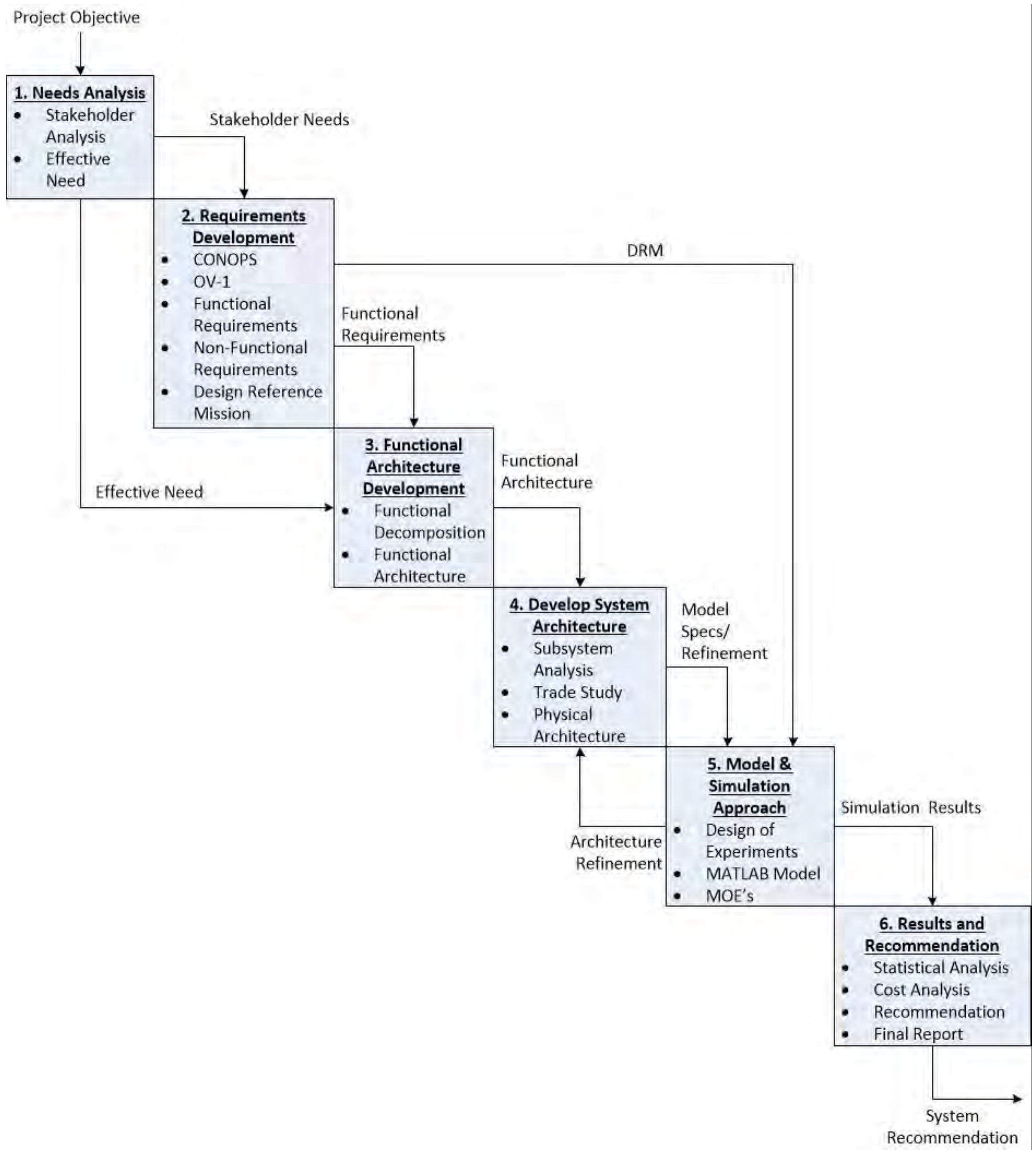


Figure 3. Project SE Process Overview

This figure depicts the customized Systems Engineering process used to develop the system recommendation. The arrows between blocks represent outputs from one phase to another. The overall model is sequential, but processes inside blocks may occur simultaneously.

The needs analysis built upon the general objective by defining stakeholder requirements and generating an effective need that could be met with a systems solution. A stakeholder analysis was performed, and the resulting stakeholders were categorized according to the type of role filled, such as regulatory or end user. Additionally, key stakeholders were identified, consisting of the entities that would be adopting the system, operating the system, or materially affected by the system. The needs of the stakeholders were discovered and used as the basis for the requirements definition phase. The phase was then concluded by deriving an effective need that could be satisfied with a systems solution.

The second phase consisted of developing system requirements. Stakeholder needs were allocated to system requirements that provided aspects of the system. These requirements were then sorted into functional and non-functional requirements. The functional requirements were used to develop a functional architecture in the fourth phase. Since the problem had been narrowed down to specific requirements, a Design Reference Mission (DRM) was developed to generate a baseline pirate boarding threat scenario, intended for use during the construction of the model. Additionally, a Concept of Operations (CONOPS) was created to allow the project team to understand the system at the top level.

The third phase developed the functional architecture of the system. A top-level function was generated from the effective need statement, and a functional decomposition was performed to derive lower level functions. Requirements were mapped to the generated functions; each function and requirement block was paired with at least one other block to avoid purchasing unnecessary capabilities and to ensure all stakeholder needs were met. The combination of the top-level and detailed functions formed the functional architecture, which was used in the system architecture development process. The phase wrapped up by developing an operational view OV-1 diagram to visualize top-level architecture.

The fourth phase developed the physical system architecture. Research was performed to identify COTS countermeasures that could provide at least one of the functional requirements of the system. The available countermeasures were narrowed to a more workable list by performing a pairwise comparison and decision matrix analysis. Weighted functional requirements and factors such as estimated cost, operational availability, and required maintenance were used to select five system configurations for analysis. The cost and logistical footprint-centric approach was used as there was little or no data on the effectiveness of the countermeasures in the field; the initial decision matrix filtered out logistically infeasible system configurations. The functional attributes of the selected countermeasures were then used in the generation of a system model.

The fifth phase consisted of building and running a system model. The operational constraints and system configurations were based on the DRM, while the success conditions were based on Key Performance Parameters (KPP) generated in the functional analysis phase. Multiple system configurations were developed and simulated to determine the effectiveness of both individual and combined countermeasures. Each scenario was iterated 1000 times and measures of effectiveness were generated for each countermeasure based on an average rate of success. A statistical analysis was performed to determine the effects of each countermeasure, and how each countermeasure interacted with other countermeasures. The highest scoring countermeasures and combinations were used to determine the system recommendation. The modeling and simulation phase fed back into the system architecture phase as the obtained results were used to refine the model and more closely reflect reality. Ultimately, the simulation results were used to rank the selected countermeasures by effectiveness.

Finally, a recommended system configuration was chosen. In order to do this, a detailed cost analysis was performed for each high scoring countermeasure due to the importance of marketing to the largest possible customer base, ranging from large corporations to independent mariners. Life-cycle costs (LCC) and required maintenance were analyzed to generate a total system cost used to rank the top countermeasure. A recommendation was then chosen based on the best value system, which contained the intersection of both cost and performance data.

B. NEEDS ANALYSIS

The second phase of the project was to perform a needs analysis to determine and refine user and customer needs, considered to be unmet capabilities or approaches that can be made more cost-effective. A Stakeholder Analysis was performed to identify entities that would be utilizing and maintaining the system, or who represent sources of technical or financial information. The results of the Stakeholder Analysis were sorted into two groups: key stakeholders, and non-key stakeholders, based on a requisite authority to dictate system needs during development. The inputs from each key stakeholder were compiled to develop an objective list of stakeholder desires. Each item on this list was then marked as either a need or a want based on the relative level of importance to the project objective.

Once a list of stakeholders had been compiled, stakeholder feedback was solicited to gain insight into each entity's needs and desires. Phone discussions were conducted with representatives from shipping companies Maersk and Cosco, a U.S. Navy officer previously assigned to anti-piracy duties, and a merchant insurance company. A list of stakeholder wants and needs was generated from the gathered input, and common themes determined. Additional correspondence was conducted with technical representatives from companies producing anti-piracy products, providing technical and cost data, as well as a unique perspective on what attributes were needed for defensive system to be a success. Additionally, an Effective Need was derived from the list of stakeholder needs that represented the broadest possible need that must be fulfilled for the system to be successful.

C. REQUIREMENTS DEVELOPMENT

The Requirements Analysis section transformed the unknown nature of the pirate danger into quantifiable criterion capable of being modeled using the results from the Needs Analysis. The top-level Effective Need was used to derive a top-level system requirement, which specifies the overall system attribute or capability required for mission success. The needs from the key stakeholders were used to generate specific requirements that would ensure the system fulfilled the associated need. Each requirement was then categorized as either functional or non-functional. A functional requirement consists of a system requirement to perform some action or function, while a nonfunctional requirement consists of possessing some attribute or quality. Each functional requirement was used to develop a comprehensive requirements hierarchy by breaking down the top-level requirement into the lower-level requirements and showing the resulting connection between the low-level requirements and overall mission success. The next step in this phase was to develop a Design Reference Mission to define a baseline threat scenario. The Requirements Development phase concluded with the construction of a top-level Concept of Operations diagram and a system-level OV-1 diagram, both used to present high-level goals and process statements in an easily comprehensible pictorial format.

The DRM allowed the team to develop a common reference point with which to analyze piracy as a maritime threat. According to Lilly, a DRM "defines the specific projected threat and operating environment baseline for a given force element … and is primarily an engineering/design tool to support systems engineering activities by identifying significant design-driving operational elements and characterizing them to the level of detail necessary to assess design impact" (Lilly 2003, 257). Once the DRM was completed, the team was able to discuss, research, and develop the operational requirements from the same vantage point as well as properly assess the feasibility of possible solutions with respect to schedule, cost, and technical maturity.

D. FUNCTIONAL ARCHITECTURE DEVELOPMENT

The Functional Architecture phase translated the system requirements hierarchy into functions, which are considered to be actions or processes that the system performs. A function is derived from each requirement by determining what action is required to meet the associated requirement. Once all requirements had been met by a function, the functional relationship was mapped by decomposing the derived functions into detailed low-level functions, which represent specific processes which cumulatively lead to mission success. Each function block fulfilled at least one requirement block, while each requirement block was met with at least one function block, avoiding unmet requirements and non-required capabilities.

The top-level through low-level functional blocks combined to form the functional architecture, which was used to construct the system architecture. Additionally, a data model was generated, which is discussed further in section F, Modeling and Simulation, and is detailed in Appendix C.

E. SYSTEM ARCHITECTURE DEVELOPMENT

The System Architecture phase allocated physical components to the functional architecture. This process was conducted by identifying physical countermeasures that would perform the desired function, and thus aid in meeting the overall functional requirement. Because this was an effort in identifying rather than developing countermeasures, our approach to developing the system architecture was to specify how multiple countermeasures could be incorporated into a common anti-boarding system and to identify Commercial-off-the-Shelf (COTS) countermeasures that would fit into the architecture. A list of suitable countermeasures was compiled, and a trade study was performed via a decision matrix to narrow the available pool of possibilities to a small, manageable list. Measures of Performance (MOPs) were created and ranked using a pairwise comparison to determine the relative importance of each factor; each factor was then weighted per the results to allow assessment of the identified countermeasures. Once the factors had been weighted, a decision matrix was constructed to score each countermeasure against how well it performed each weighted factor, and the resulting scores were sorted highest to lowest. The top-performing countermeasures were selected for modeling and simulation to determine which system-of-systems (SoS) configuration of countermeasures offered the best anti-piracy performance.

Research indicated that most commercial vessels do not deploy pirate countermeasures; amongst vessels that did, typically one countermeasure was used. The commercially available countermeasures were sorted into categories based on the expected amount of crew management required to operate the system during a pirate attack:

- passive defense, a countermeasure that requires no crew management once deployed and affects pirates attempting to board the target vessel.

- active defense, a countermeasure that requires crew management when in operation and affects pirates attempting to board the target vessel.
- active offense, a countermeasure that requires crew management when in operation and can affect pirates and/or pirate craft at range.

Several Measures of Performance (MOP) were developed to analyze each piracy prevention countermeasure. Each measure was weighted using a pairwise comparison, which determines the relative importance of each measure and allows multiple countermeasures to be ranked according to determine decision-making criteria. A full list of the MOP's is listed below along with a description and the evaluation criteria.

Measures of Performance

The MOPs listed below were used for an initial evaluation of the COTS countermeasures to narrow the field of candidate countermeasures by and determine which of them were worth investigating.

- **Time to deploy**
    - Deployment time is based upon the average length of time required for an average crewmember to set up the anti-piracy defensive system
    - Rating of 1 for < 1 minute to deploy
    - Rating of 5 for >5 minutes to deploy
- **Ease of use**
    - This measure was based on the usability of the option by an average crewmember, relating to its effectiveness of use
    - Rating of 1 for few steps and intuitive use
    - Rating of 5 for numerous complex steps requiring skill and training
- **Maintenance**
    - This measure was based on the need for and length of maintenance required for the anti-piracy option to be effective over a year of average use. This related to number of parts and complexity of the parts.
    - A rating of 1 was used for simple maintenance which could be performed onboard the vessel with little training by an average crewmember
    - A rating of 5 was used for options which require specialized offsite maintenance or calibration by the OEM

- **Cost**
    - This measure was based on the cost of use over the first five years of the options deployment, including purchase, operations, maintenance, and training costs. Rough estimates for cost were used at this stage due to the number of options.
    - A rating of 1 was given for an option costing less than $10,000 in its first five years of deployment
    - A rating of 5 was given to an option costing greater than $250,000 in its first five years of deployment
- **Ease of overcoming**
    - This measure was based on the difficulty with which the adversary would have overcoming the countermeasure and continuing with their attempt to gain control of the vessel
    - A rating of 1 was given to options which required high skill and or numerous steps to overcome
    - A rating of 5 was given to options which required little skill or number of steps to overcome
- **Need for logistics support**
    - This measure was based on the need to support the anti-piracy option with communications, intelligence and or consumables
    - A rating of 1 was given to options requiring little logistics support
    - A rating of 5 was given to options requiring high levels of logistics support
- **Effect on Crew**
    - This measure was based on the effect of the countermeasure on the crew onboard the vessel using the option. Some options utilized acoustic or visual methods which could possibly harm the users if they malfunctioned slightly.
    - A rating of 1 was given to options which had a low probability of harming or incapacitating the crew of the vessel using the option
    - A rating of 5 was given to options which had a high probability of harming the crew of the vessel using the option

Each measure was assigned a raw rank from 1 to 5 to designate relative importance to the system. The weighted level was determined by developing a linear equation containing the summation of the rank multiplied by a variable x, where to total of all weights is equal to one. Once the value of x was determined, it was multiplied by the raw rank to determine a relative weighted value of each measure. The results of pairwise comparison can be found in Table 3, and were used in a decision matrix ranking system.

Because the purchasers of the system were expected to be highly sensitive to the cost of the final selected system, this criterion was added to the decision matrix as a weighted measure. This allowed cost to be considered during the process of selecting the countermeasures to be modeled, reducing the likelihood that all selected countermeasures would have an unacceptable cost/benefit ratio when the simulations were complete.

Table 3. Measures of Performance Criteria Ranking Weights

This table shows the weighting of MOPs by assigning a raw rank from 1 (low) to 5 (high) to each measure, then normalizing rank by dividing each MOP by the sum of raw ranks to obtain a percentage of total possible rank.

MOP	Raw Rank	Reasoning for Raw Rank	Normalized Rank Weight
Time to deploy	3	Countermeasure must respond quickly to unexpected attack	0.1305
Ease of Use	3	Countermeasure must be easy to use effectively due to high stress situations	0.1305
Maintenance	3	Countermeasure must not require excessive maintenance to minimize Operations and Support costs.	0.1305
Cost	5	Countermeasure must represent low cost/benefit ratio for purchase cost to encourage widespread adoption	0.2175
Ease of overcoming	3	Countermeasure must resist pirate attacks	0.1305
Need for Logistics Support	1	Countermeasure must avoid encumbering the crew or associated ships with excessive logistics support. Common consumables can be purchased at friendly ports.	0.0435
Effect on Crew	5	Countermeasure must maintain or improve the safety of the crew during a pirate attack.	0.2175

The decision matrix scores were based on the convention that a rating of 1 was desired and a rating of 5 was undesirable. The scores, based on the ratings of the countermeasures multiplied by the rankings of the measures, were summed across all measures then ranked based on their proximity to 1 or 5. The sum of the weighted scores was then normalized to a percentage, representing how well each countermeasure met the ranked factors. The results for the examined system configurations are shown in Section III. The five highest ranked options were chosen for further analysis and input into the modeling phase.

F. MODELING AND SIMULATION APPROACH

The Modeling and Simulation Approach consisted of constructing and running the system model. The model utilized the DRM to generate the operational constraints and deterrent, while the only success condition was the commercial vessel preventing a successful boarding; the result was a model that accurately portrayed the operational environment. MATLAB, a programming language and computational environment, was used to create the model and run the associated simulations. The central variables contained a mix of random, constant, and probabilistic factors:

- pirate skiffs were initially set at a constant range from the commercial ship
- number of skiffs and initial angular direction from commercial ship were randomly chosen
- countermeasure strikes for targets at range were assigned a probabilistic value of hitting the target.

Multiple system configurations were developed and simulated to determine the effectiveness of both individual and combined countermeasures. Each scenario was run 1000 times and measures of effectiveness (MoEs) were generated for each countermeasure based on an average rate of success.

The randomized design of experiments (DOE) was constructed to ensure the generation of valid data and efficiently plan the scenario run order. The top five countermeasures from the decision matrix each contained two discrete states, on and off. Thirty-two possible system configurations were predicted using formulas (1) and (2) below:

$$\left(possible\, scenarios\right) = \left(number\, of\ states\right)^{\left(number\ of\ countermeasures\right)} \quad (1)$$

$$\left(possible\, scenarios\right) = 2^5 = 32 \quad (2)$$

The number of system configurations was reduced to twenty-four by removing the eight in which neither of the passive defense system configurations was used, despite an active system being utilized. An active defensive system is more manpower intensive than a passive system, and must be operated immediately before or during a pirate attack. The large cost difference between a passive system and an active system indicates that utilizing an active system without a passive component would be considered gold-plating, and as such should not be considered for recommendation. A practical scenario can be pictured to further illustrate this reduction: it is not standard practice to protect a home with a high-quality security system, yet neglect to install a lock on the door. As such, the number of possible system configurations can be safely reduced without filtering out potentially valid solutions.

A statistical analysis was performed to determine the effects of each countermeasure, and how each countermeasure interacted with other countermeasure. The highest scoring countermeasure and combinations were used to determine the system recommendation. The modeling and simulation phase feeds back into the system architecture phase as the physical architecture, and thus the model, utilizes the obtained results to refine the model and more closely reflect reality.

The model was set up to determine the percentage of successful defenses against pirate attacks through use of passive defense, active defense and/or active offensive methods for the status quo scenario as well as finding the best current solution and recommending options for further advanced study.

Model system configurations were run against a DRM with varying piracy countermeasure combinations enabled aboard the target ship. For each configuration, 100 simulation runs were performed and evaluated and the overall percentage of successful defenses of the vessel were calculated to generate initial data in order to facilitate analysis. A follow-on effort of 1000 runs for each scenario was completed to validate and refine the initial results by filtering outlying values and generating a precise average. The cost of each scenario was independently calculated, and the success percentage and cost were equally weighted against each other to determine the best scenario.

1. Model Foundation

The Model was created as an incremental-time object-oriented MATLAB program. MATLAB was selected due to its wide availability, commonality across organizations, and flexibility. Lower-level programming languages, which are close to machine code, were declined due to the reduced ability to transfer the model amongst different computing environments. Higher-level modeling tools, which are closer to human spoken language, were not selected due to concerns over their flexibility and reduced availability. MATLAB represents a midlevel programming language that maintains most of the flexibility of the lower-level languages, while using the higher-level language verbiage.

2. Model Architecture

Figure 4 shows the overall architecture structure of the model. The program was divided into thirteen object classes, three enumeration classes, and one helper function file. The enumeration classes are used to define commonly used states for other classes and map logical states to discrete values for MATLAB to track. The helper function is used to define the commonly used get_angle function as it does not inherently belong to any particular class. The object classes define the logical constructs that the model is built of and are described below. Additionally, a UML relational diagram was generated to control and document the interactions between the model functions. This diagram can be found in Appendix C.

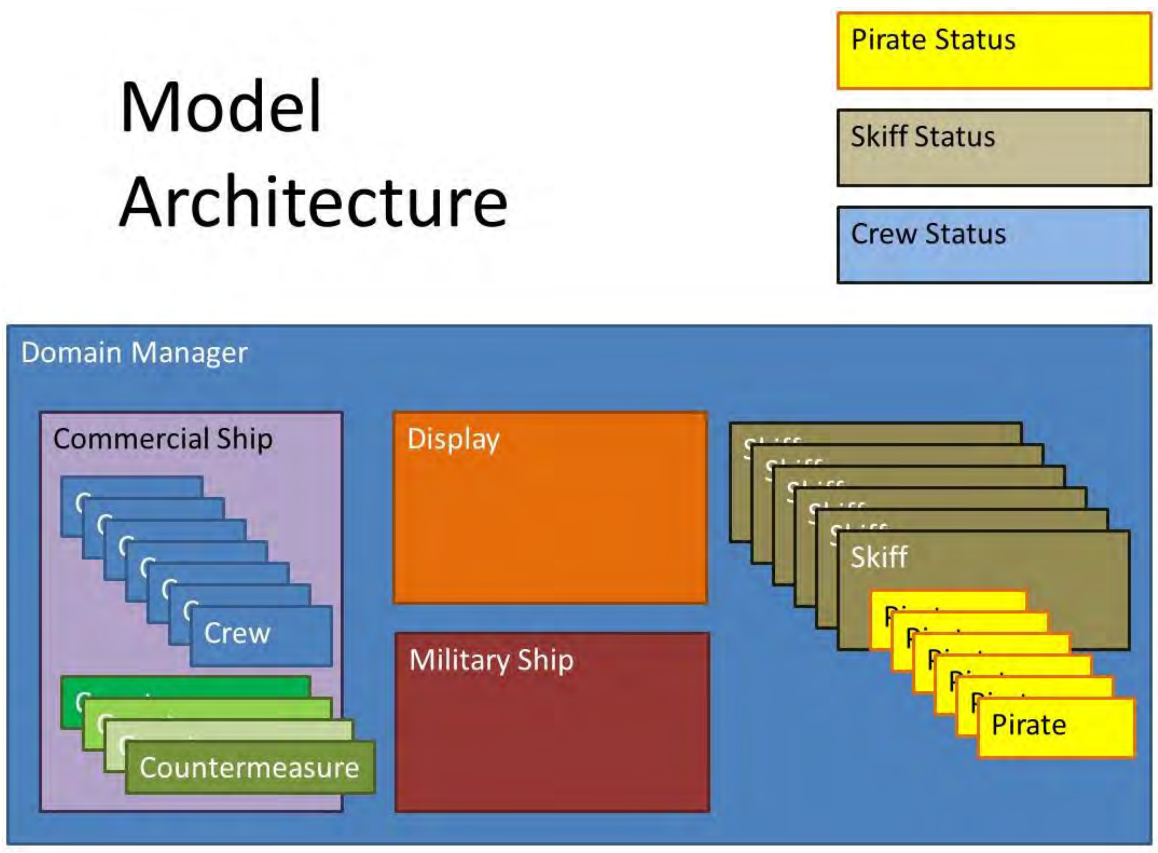


Figure 4. Model Architecture

This figure depicts the Domain Manager and the modules created inside it, consisting of Commercial Ship, Display, Military Ship, and multiple instances of pirate skiffs. The commercial ship and the skiff modules each create low level modules that track either crew or pirates, while the commercial ship also creates countermeasure modules.

a. *Domain_Manager*

The prebuilt functions included in MATLAB are not powerful enough to manage a model of this complexity, so a custom background object called Domain Manager, was developed to control operation of the overall model. It creates the other objects within itself, initializes them, and then calls their operations iteratively to simulate the passage of time. Domain Manager controls the passing of data between the other objects (as opposed to a shared memory structure). Domain Manager determines when a scenario has concluded based upon established criteria; primarily that all pirates have been disabled or that one has boarded. The Domain Manager also controls repeated runs of the various system configurations in order to calculate the overall MOE. Domain Manager is the connection between the MATLAB user interface and the rest of the program, it takes in the scenario configuration and outputs the calculate probability of survival for the commercial ship.

b. Com_Ship

The Commercial Ship class represents the target ship from the DRM within the model. It tracks the location, velocity and all other necessary factors associated with the ship's status and actions. The Commercial Ship object contains within itself objects representing all of its crew members as well as the objects associated with each countermeasure.

The Commercial Ship is represented within the model not as a rectangle but rather as a point in space. Figure 5 shows how the dimensions of the ship sides were approximated with angles from the center point.

Note from Figure 4 that the Crew objects are shown in the same color but the Countermeasure objects are varying. This is because Commercial Ship creates several instances of the same Crew class but the Countermeasures actually consist of various class objects that merely have similar roles in the architecture.

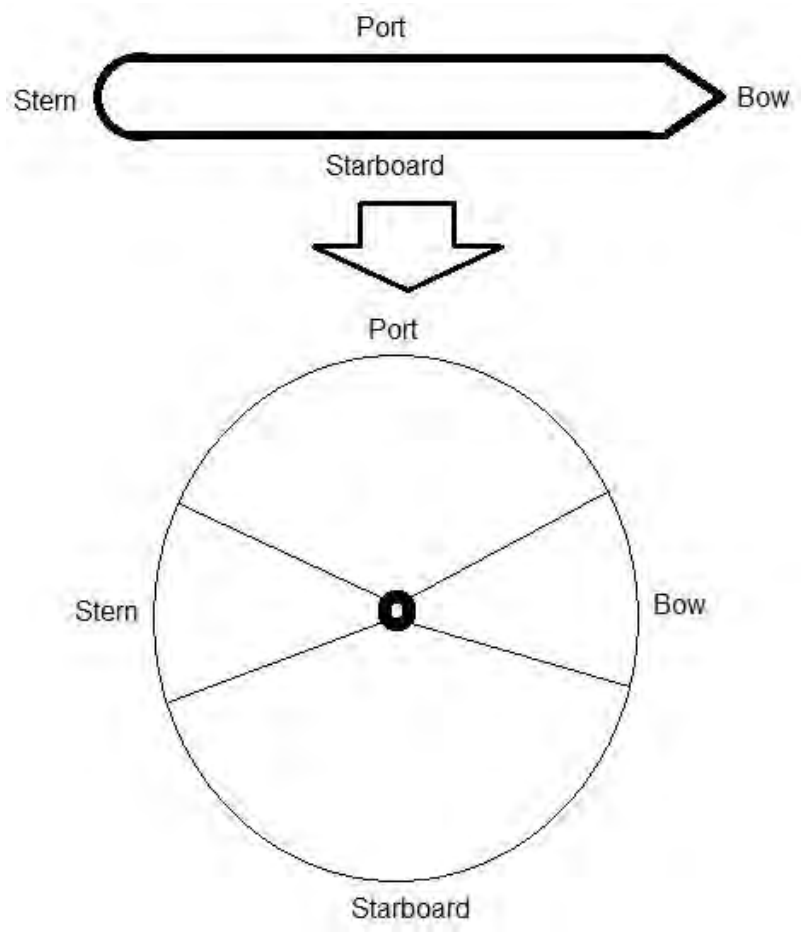


Figure 5. Com_Ship Angular Approximations

This figure depicts how the dimensions of the ship sides were approximated with angles measured from the center point.

c. Mil_Ship

The Military Ship class generically represents some assisting vessel coming to the aid of the Commercial Ship that can stop the pirate attack if it arrives in time. The program object actually does very little, merely tracking its own progression. If it reaches the Commercial Ship then Domain Manager will end the scenario.

d. Skiff

The Skiff class represents the pirate vessels within the model. The object tracks the status (active or disabled), position, and remaining crew of a particular skiff and determines the skiff's next actions when called. As shown in Figure 4, each Skiff object also contains a number of Pirate objects (described below) and triggers them to carry out their own actions in each time increment. The Skiff objects are all generated and tracked by Domain Manager.

In general, Skiffs move directly towards the Commercial Ship in an attempt to allow their Pirates to board. Skiffs can be disabled by countermeasures, rendering them immobile. If all Skiffs in the scenario are disabled the Domain Manager will end the scenario.

e. Display

The Display class is a special use class that is only invoked when a demonstration version of the model is called from Domain Manager. The Display object translates the data stored within Domain Manager into a visual display of the moving ships and skiffs for each time increment. Display was created primarily to enable debugging, validation of overall operation, and to facilitate presentation and explanation of the model. Figure 6 shows an example of the Display class output. Note that active Skiffs are denoted by red circles, disabled ones by grey, the Commercial Ship by blue and the Military Ship (out of view in this case) by green.

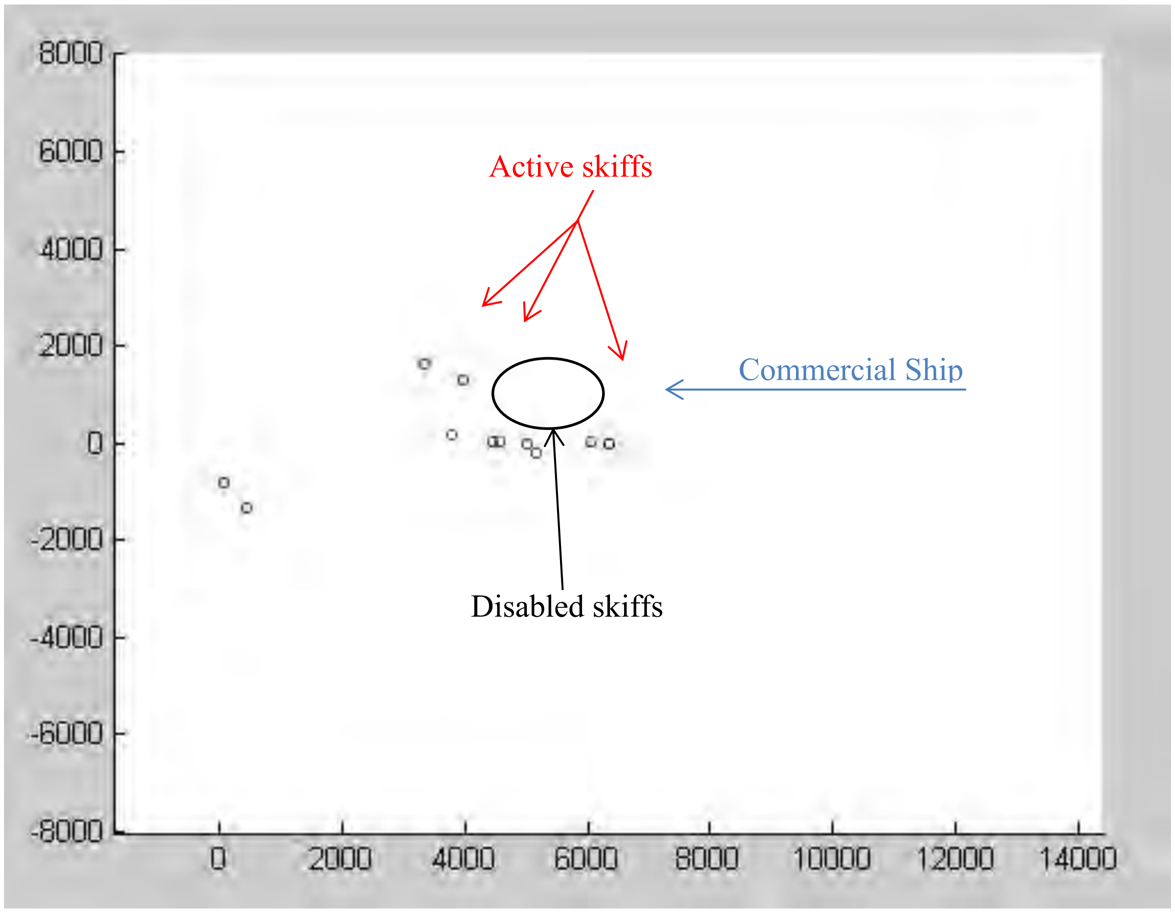


Figure 6. Example of Display

This figure depicts the display module tracking the commercial ship, disabled pirate skiffs, and active pirate skiffs. Skiffs begin as red circles and turn black if neutralized. The commercial ship attempts to approach military vessel (not shown) before being overwhelmed.

f. Pirate

The Pirate object represents an individual pirate. A Pirate can transition primarily between different states and keeps track of the time required for individual pirates to accomplish tasks, such as boarding the side of a Commercial Ship.

Pirate objects are created and reside within Skiff objects. Seven pirate objects are created at the start of each run. The Skiff passes update calls down to the Pirate object as well as all data or pointers needed for it to accomplish its tasks.

g. Crew

The Crew class is largely a vestigial remnant within the model. Most of the intended functionality of the class was de-scoped from this iteration of the model. Crew objects reside within the Commercial Ship object.

h. Pirate_Status/Skiff_Status/Crew_Status

The three Status classes define enumeration sets. Enumeration sets are used in programming to provide meaningful names in software code to what is essentially a list of identifiers. In this example each of these classes represents a list of possible states that their respective objects may be in at a given time. The enumerations can be invoked throughout the entire program to provide a meaningful title to the status as opposed to a number.

i. Countermeasures

The Countermeasure classes are each defined individually. They were generated this way, instead of through an inheritance structure, due to the drastically different ways in which the countermeasures operate. Additionally, different types and fidelity of data are available for the different countermeasures and it was not prudent to treat them similarly. Five shipboard countermeasures were selected from the trade study for modeling and simulation. The approach used to model the individual Countermeasure classes is described in the following chapter.

3. Countermeasure Implementation

a. Pirate Trap (P-Trap)

The pirate trap countermeasure is a system of difficult to see lines trailed through the water along the sides of and behind the commercial ship in order to foul the propellers of pirate craft. Figure 7 shows how these physical regions were translated into angular representations within the model.

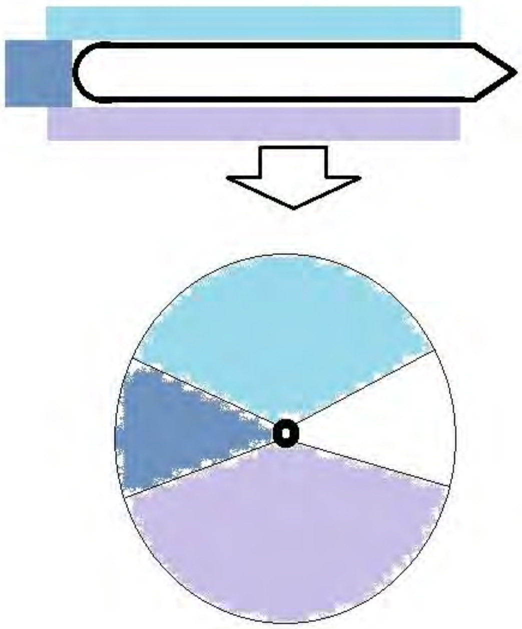


Figure 7. P-Trap Regions of Effect

This figure depicts how the trailing lines of the p-trap were translated into angular representations within the model.

Each P-Trap region within the model can stop 10 pirate skiffs in a given scenario. Each region is created as a separate instance of the class and tracks how many lines it has remaining.

a. *Water Cannon*

The water cannon countermeasure features a remote-controlled water turret that operates like a fire hose in suppressing and forcing away pirates. The intended use is to flood the skiffs, but it was determined that the pirates would seek to avoid this eventuality so within the model the Water Cannon object acts to force skiffs out of its range. The Water Cannons act on one skiff at a time and causes them to flee the Commercial Ship's proximity. Each of the six Water Cannons is created as its own instance and tracks its own tasking. The Water Cannon Regions are shown in Figure 8.

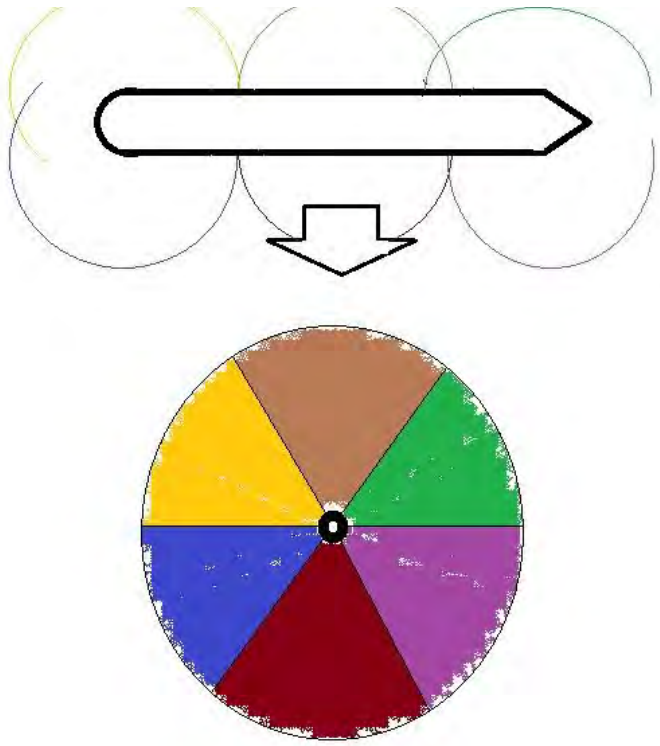


Figure 8. Water Cannon Regions of Effect

This figure depicts placement of water cannons on ship, as well as the associated angular representation used in the model.

b. Razor/Barbed Wire

The countermeasure option of wrapping the perimeter of the ship with barbed or razor wire is represented by the Barbed Wire class. The Barbed Wire object in turn creates a large quantity of Barbed Wire Segment objects which each track the health of the barbed wire over a small portion of the Commercial Ship's circumference.

The Barbed Wire directly adds two minutes of scenario time needed for a Pirate to bypass the Barbed Wire Segment. Multiple Pirates at the same Barbed Wire Segment can work together to accelerate the time they bypass it in.

c. Pirate Curtain

The commercially advertised pirate curtain system consists of a combination of fire hoses used to flood pirate skiffs and erratically flailing hoses with weighted ends that can cause bodily harm to individuals scaling the side of the vessel. It was determined that the first component of the system heavily overlapped with the Water Cannon already under consideration, but that the flail version represented a unique countermeasure option. The Pirate Curtain class then represents the flail portion alone of the commercially proposed solution.

The Object monitors the Port and Starboard regions of the Commercial Ship and applies a chance to strike any pirate who is in the process of attempting to board. If struck, the Pirate is presumed to be permanently disabled within the timeline of the scenario.

d. Air Cannon

The air cannon is a mounted, remote controlled turret that fires one of several projectiles to stop pirate skiffs. Selected for the model from among these was the net/line option that is fired at pirate craft to ensnare their propellers.

The Air Cannon class fires nets at regular intervals at random pirate skiffs within its range. With each shot there is a chance to miss. The cannon is currently represented as having a clear field of view across all angles. It has a limited number of shots for each scenario based off the assumption that pirate weapon fire will prevent the crew from reloading the deck-mounted launcher.

4. Model Implementation

Appendix B contains the model script code and soft copy is available upon request. Appendix C contains a Unified Modeling Language (UML) diagram showing how the class relationships were implemented.

G. COST ANALYSIS APPROACH

Commercial shipping is a profit-driven industry, and as such will only adopt a piracy defensive system if the benefits outweigh the cost of implementation. The system will be marketed towards a range of merchant companies, ranging from independent ship-owners running single ships to mega-corporations such as Maersk, who command over 600 vessels. A cost analysis was performed on all system configurations for factors such as initial purchase cost as well as logistical support, and the results combined into a five year total cost.

1. Cost Model

The cost model used in this project for each countermeasure was formulated based on a number of factors specific to each system. The cost was calculated as a five year total ownership cost using net present value. An aggressive, 6% yearly inflation rate was used to reduce the likelihood that the estimated costs generated by the model would underestimate the Boarding Prevention System cost in order to account for items not included in the model. Since the shipping companies likely have large bank accounts, the interest that the companies earn on their product or overhead accounts was estimated at 3.3%. The timespan of five years was used as that is a typical overhaul time period for U.S. naval vessels, at which point the ships maintenance authority would utilize an overhaul budget separate from the O&M budget used for normal operations.

Manufacturing companies of current anti-piracy systems fitting our system descriptions were polled to determine system specific cost information. The companies were asked to provide data on initial purchases, maintenance, IT support, integration cost and operation specific data for each system. Data was reported as months for time based data and FY14 dollars for monetary data. The requested data included:

Initial Purchase

Estimated Initial Contracting percentage
Initial Consumables price
Initial System Price
Manufacturer
Number of Systems needed to support vessel

Maintenance

Estimated time between unplanned maintenance
Estimated cost for unplanned maintenance
Estimated Routine maintenance cost
Estimated time between routine maintenance

IT
Estimated interval between IT support
Estimated IT Support cost

Complexity
Estimated ship integration cost per system

Operations
Estimated system life
Estimated routine Consumable cost
Estimated Time between Consumables purchases

2. Labor Cost

The labor cost of operations for vessel crew members was not factored into this cost estimate as the labor rates per shipping company representatives may vary based on a wide variety of factors including time of year, national origin of the shipping company, etc. The contracting effort was taken to be a series of one time purchases with a single contracting percentage. More complex contracting vehicles, such as those with multi-year options or clauses or variable contracting percentages, are out of the scope of this project.

3. System Purchasing

Total purchasing price for the vessel-wide Boarding-Prevention System was separated into sub-categories as well as the initial purchase of materials or services prior to the initial build and integration as well as for five years of system use. A limitation of this model is that it assumes that the boarding-prevention countermeasures will be used on a scheduled basis, which will hopefully not be the case. As this will likely assume usage on a higher rate than real world usage for these systems, the estimate will be slightly high, thus deferring unforeseen charges. Year zero is taken to be FFY14 and accounts for the vessel being in a maintenance status for a lengthy repair time period. During the time period in year zero the vessel holding company will contract for the purchase, integration documentation and crew training regarding the vessel-wide boarding-prevention system and perform the purchasing as well as the integration efforts. A short time frame is sufficient for contracting efforts of this type of system since the boarding-prevention systems will be COTS items with minimal modifications. Year zero costs therefore are in FY14 costs and Year one will start as of FY15 for actual countermeasure usage. The consumables and maintenance budgeted for year zero will be utilized for the initial integration effort and those line items budgeted for years one through five will be used for underway use.

The total initial purchase price is based on formula (3) below:

$$\begin{aligned}&\textbf{Initial purchase}(\textbf{IP}) = (1+\textit{contracting percentage})* \\ &(\textit{countermeasure price}+\textit{consumables price})*(\#\textit{of countermeasures})\end{aligned} \tag{3}$$

4. System Life-Cycle Considerations

Since the system is estimated to have a usage life and not last indefinitely, the estimated countermeasure life (assuming regular maintenance per the manufacturer) was taken into account for system purchases in out years. A round-down function noting the number of months for the countermeasure life span as well as the number of months until the end of the purchase year in question was used to predict the system purchase price for all out years up until five years have passed. The year zero is taken to be the initial purchasing contract pricing, with years one through five being the O&M budget for the ship for each of those years, regarding the boarding-prevention countermeasure usage. The round-down function, executed via Excel, accounts for the fact that each purchase will be an individual event and partial system purchases will not be made.

System purchase in years one through five is based on formula (4):

$$SP_n = IP*ROUNDDOWN\left(\frac{12*(\textit{year } n)}{(\textit{system life})},0\right)-SP_{(n-1)} \tag{4}$$

The total integration cost (TIC) is based on the following formula (5):

$$TIC=(1+\textit{contracting percentage})*(\#\textit{of systems})*(\textit{system integration cost}) \tag{5}$$

For years one through five, a round-down function was used to determine the number of countermeasures needing to be integrated for each of those years. It takes into account the system life as well as the number of months which have elapsed by the end of that fiscal year (FY) and the total system integration cost. Additionally, the round-down function takes factors in that the system integrations will be individual events and cannot happen as partial events.

System Integration for out years is based on the formula below (6):

$$SI_n = TIC * TIC * ROUNDDOWN\left(\frac{12*(year\, n)}{(system\, life)}, 0\right) - SI_{(n-1)} \tag{6}$$

5. Consumables

Consumables used in the total system under routine use were accounted for in a yearly consumables line item. The routine consumable cost for a set time span for each individual system was obtained from the manufacturer as well as the length in months of that time span, i.e., how often a batch of consumables would need to be purchased. Year zero routine consumable purchases represent the price for one set of the routine consumables in addition to those which will come with the countermeasure, since those often become expended during check out testing. For years one through five the pricing is calculated using a round-down function (7) including the routine cost, routine usage period and the total number of months until the end of that yearly time period; the costs associated with previous years are subtracted. The round-down function accounts for the fact that the consumable purchases will occur as individual events and cannot occur as partial events.

$$Consumable\, Purchase\, for\, years\, 1-5\ (CP_n) = (routine\, consumable\, price) * \\ ROUNDDOWN\left(\frac{12*(year\, n)}{(estimated\, time\, between\, routine\, replacement)}, 0\right) - CP_{(n-1)} \tag{7}$$

6. Maintenance

Routine maintenance costs are calculated by accounting for the time period in question (initial or later years), as well as the mean time between maintenance for the countermeasure and the routine maintenance costs. As with other cost sub-categories in this cost model, the routine maintenance cost uses a round-down function (8) which accounts for each routine maintenance action being an individual action which cannot be conducted as a partial action. For years two through five the maintenance performed in previous years is subtracted out in order to not charge for a maintenance action multiple times.

$$\begin{aligned}&Routine\,Maintenance\,Cost\;for\;years\,1-5\;(RM_n)=\left(estimated\;routine\;maintence\;cost\right)*\\&ROUNDDOWN\left(\frac{12*\left(year\;n\right)}{\left(mean\;time\;between\;routine\;maintenance\right)},0\right)-RM_{(n-1)}\end{aligned} \tag{8}$$

Unscheduled maintenance cost is calculated by assuming a ratio of routine to unscheduled maintenance. Or the total maintenance period, it is estimated that 78 % of the maintenance is routine maintenance since these systems are only a few levels deep regarding systems of systems. This makes the unscheduled maintenance 22 % of the total number of maintenance actions. If a particular system has reason to believe that the ratio between routine and unscheduled maintenance is something different than this, the ratio of routine to unscheduled maintenance is a variable which can be changed within the model. The time between unscheduled maintenance events is taken from the ratio of unscheduled to routine maintenance actions as well as the MTBM for routine maintenance (9). This MTBM for unscheduled maintenance is used along with an estimated cost for the unscheduled maintenance actions to determine the yearly unscheduled maintenance costs. A round-down function (10) is used to determine the number of individual unscheduled maintenance actions within the yearly time frame; this round down function takes into account the fact that the unscheduled maintenance actions are individual actions which can’t be performed as partial events. If a cost for unscheduled maintenance was unable to be found from a vendor, the cost for the unscheduled maintenance was estimated to be three times the cost of scheduled maintenance for that same system.

$$Mean\,Time\,Between\,Unscheduled\,Maintenance\left(MTBM_U\right)=\frac{MTBM_R}{0.28205} \tag{9}$$

$$Unscheduled\,Maintenance\,Cost\,for\,years\,1-5\left(UM_n\right)=$$
$$\left(estimated\,unscheduled\,maintenance\,cost\right)*ROUNDDOWN\left(\frac{12*\left(year\,n\right)}{MTBM_U},0\right)-UM_{(n-1)} \tag{10}$$

7. IT Support

After detailed queries for each countermeasure had been performed, it was determined that the articles onboard ship would not likely have automated tracking or self-diagnostic systems and that IT support would not be needed. In this case, the values within the algorithm were set to zero for IT support (11).

$$IT\,support\,cost\,for\,years\,0-5\left(IT_n\right)=0 \tag{11}$$

8. Documentation

Documentation is a key piece of the operating environment for the countermeasures. In order to perform the initial documentation effort, it was assumed that a quality assurance representative from either the manufacturing company or the vessel holding company would spend 80 hours' time (two week time frame was assumed based on team member experience with documentation efforts) drafting the initial documentation manual, after which a team of senior engineers would spend approximately three weeks' time to review and publish the documentation. Fully burdened labor rates for quality assurance representatives and senior engineers were taken from the department of labor website at \$89.19 and \$113.08, respectively. Given that countermeasures with more complex sub-systems would require more documentation for use and maintenance of those sub-systems, the complexity factor discussed above was multiplied against the sum product of the labor rates and time required by the QA representative and the senior engineers. For later years it was assumed that the documentation would need to be updated approximately once a year based on team member experience with IT software documentation updates for the Navy Oil Analysis Laboratory or NOAP program. The cost for these documentation updates was estimated to be half of the cost of the initial documentation effort, based on half the number of labor hours being needed.

The hourly rates consist of :

- Hourly Rate, Quality Assurance Representative (HRQ) = $89.19
- Hourly Rate, Senior Engineer (HRS) = $113.08

Formulas (12) and (13) were used to determine initial and recurring documentation costs:

$$\textit{Initial Documentation Cost}\left(IDC\right)=\left(80*HRQ+120*HRS\right) \tag{12}$$

$$\textit{Documentation for years}\,1-5\left(DC_n\right)=0.5*IDC \tag{13}$$

9. Training

Training for each countermeasure was assumed to be performed during the initial build/integration/trial period. The quality assurance representative, who wrote the manual, or an equally paid and competent contemporary, would perform the training for the units for the entire vessel staff. It was assumed that the training would take two weeks (80 hours) worth of time at the quality assurance representative's labor rate, as discussed above (14). This time was also multiplied by the complexity factor to account for countermeasures, which may have lengthy training needed to train crew in proper use (15). IT was assumed that the training would only need to occur once, and any further training would be passed down from crew member to crew member on the vessel, and therefore does not need to be accounted for within the cost estimate. Travel for the quality assurance representative was not accounted for within the cost estimate as the manufacturing company location and the vessel location are unknown.

$$\textit{Initial Training Cost}\left(ITC\right)=80*HRQ \tag{14}$$

$$\textit{Training cost for years}\,1-5\left(TC_n\right)=0 \tag{15}$$

10. Net-Present-Value Calculation

The cost for initial purchase of the vessel-wide countermeasure installation, the vessel-wide system integration, the routine consumable price, routine and unscheduled maintenance costs, IT support, documentation and training are all summed up each year and multiplied by the yearly estimated inflation then divided by the yearly interest generated by the vessel company to get a yearly present value for the boarding-prevention countermeasure from year zero to year five (16). The net present value was calculated by summing each yearly present value. It was assumed that the countermeasures would be used until they no longer functioned properly, and as such there was no resale price taken into account at the end of the five-year period as with some system configurations.

The variables used as inputs for all five countermeasures were based on OEM queries as well as trade articles and estimation by similarity when other values could not be found. Due to relative system simplicity, the complexity factors for each system were set to one. Routine maintenance costs were not available from the manufacturers, so a standard of two men for eight hours at $40/hr. was chosen, giving a $640 cost to all routine maintenance events. Unplanned maintenance events were estimated to take three times longer than their scheduled counterparts, therefore making them $1920 per event. An unscheduled maintenance event cost of $8,000 was found for the compressed air launcher based on manufacturers input. The MTBM for routine maintenance events was found from the manufacturer in most cases, and estimated by similarity in the case of the razor wire.

$$\begin{gathered} Present\,Value\,per\,year\left(PV_n\right) = \\ \left(SP_n + SI_n + CP_n + RM_n + UM_n + IT_n + DC_n + TC_n\right) * \frac{1.06^n}{1.033^n} \end{gathered} \tag{16}$$

NOTE: PV for year zero was calculated with the initial costs for each variable (17).

$$Net\,Present\,Value\left(NPV\right) = \sum_{n=0}^{5} PV_n \tag{17}$$

H. SELECTING THE SYSTEM SOLUTION

Shipping companies ranging from large corporations to independent mariners are expected to adopt the boarding prevention system; the final stage of this project determined the intersection point of cost vs performance to appeal to the broadest possible number of system users. Due to the profit-driven nature of the commercial shipping industry, any countermeasure must both provide an effective defense against maritime piracy and be economically affordable. The best value system was one which would be both effective for survivability in an attack situation and cost-effective.

The modeled system configurations were scored using the survival data from the modeling phase and the five-year cost from the cost analysis. All system configurations that did not produce mission successes were discarded to remove inconsequential data. Each scenario was plotted on a single chart to produce a visual representation of the results; the x-axis represented cost while the y-axis represented survival rate.

A numerical analysis was then performed to normalize cost and survival data into a single ranking. The scenario with the highest rate of success was determined, and all individual system configurations were divided by the first; the results indicate to what extent each scenario matches the performance of the theoretical maximum. A similar operation was then performed on the cost data to determine lowest cost, and how well other system configurations matched this value. The cost ranking and survival were then multiplied together to determine the overall ranking. Three solutions were selected, consisting of a cheap scenario with medium effectiveness, a high-effectiveness scenario, and a midpoint scenario.

THIS PAGE INTENTIONALLY LEFT BLANK

III. RESULTS

A. NEEDS ANALYSIS RESULTS

1. Stakeholders

By following the process outlined in the Analysis section, the list of stakeholders below was generated. All needs of the stakeholders were aggregated, but not all were implemented in creating system requirements, as some needs were not cost feasible.

a. Key Stakeholders

Our research showed that the key stakeholders were the crew that would use the BPS system, the shipping companies owning the vessels the BPS would be installed on, and the customers of said companies. The stakeholders are detailed below.

Shipping Companies

A multitude of commercial shipping companies operating vessels under many different flags (US/European shipping ~20% of traffic). The companies are responsible for implementing new boarding-prevention solutions including the cost of incorporating the solution as the primary stakeholder for the output of this project.

Merchant Vessel Crew Members

The safety of the crew is of obvious importance. Crew members that may be taken hostage have an inherent risk of being harmed or even killed if pirate demands are not met in a timely manner. Pirates may even harm the crew to prove the seriousness of their intent. Further, any component of the system solution that is an active offensive or defensive measure must be operated by a crew member.

Shipping Company Customers

The customers of shipping companies have a certain confidence in any given shipping company to deliver cargo to the destination for a cost and in a timely manner as agreed in their contracts. If a merchant vessel is captured, the cargo may be lost. Even if a captured vessel is returned to shipping company control, the cargo will almost certainly be late. Customers of shipping companies often have customers themselves in the global merchant market. Therefore, pirate deterrence is of significant important for customers of shipping companies.

a. Other Stakeholders

Pirates

The pirates, being the subject of prevention, must be considered a stakeholder of the system. It should be noted that the pirates are considered to be a dynamic force, and therefore shall respond to the system as necessary, modifying tactics to nullify or mitigate the effectiveness of countermeasures. For example, if pirates notice that the starboard side of the ship is incorporating P-Trap countermeasures, they may choose to attack the port side of the ship. Factors such as these were incorporated into the statistical probabilities of each selected component of the system's ability to deter pirate boarding.

USPACOM (US NAVY Pacific Command)

USPACOM is the United States Navy element responsible for the Pacific Ocean area, including the waters around Indonesia. It consists of the United States Third and Seventh Fleets and several other subordinate task forces. USPACOM has a potential interest in the results of this capstone project.

IMO (International Maritime Organization)

The International Maritime Organization (IMO) is a specialized agency of the United Nations whose primary responsibility is improving the safety and security of international shipping. Its secondary responsibility is environmental, preventing marine pollution from ships. The standards and regulations of the IMO apply to all vessels that operation internationally. Their stake in anti-piracy operations in the Indonesia region pertains to their responsibility to maintaining and improving the safety and security of international shipping.

Malaysian, Singaporean, and Indonesian (Local) Navies

These countries are currently providing the bulk of patrolling naval forces in the affected region. These parties have a potential interest in the changes in commercial vessel response to pirate attacks in their national waters.

2. Stakeholder Needs

The list of stakeholders and needs discovered through the research process outlined in Chapter II are listed in Table 4. Similar needs and desires were grouped together to minimize space and aid in system design, and an effective need was generated from the inputs gathered from the key stakeholders.

Some of the suggested needs are easily incorporated into the scope of a Boarding Prevention System, such as logistical support considerations. However, other inputs were not feasible as requested, such as eliminating maritime piracy worldwide. Others were not within the scope of creating a boarding prevention system, such as tracking suspicious vessels. The stakeholder needs discovered in the analysis are listed below.

Table 4. List of Identified Stakeholder Needs

This table summarizes the needs identified from the stakeholder analysis and notes which stakeholders have a given need.

Stakeholder Need	Applicable Stakeholders	Within Scope Of Project?
Provide Situational Awareness (SA)	Shipping Companies Merchant Vessel Crew	No
Track Suspicious Vessels	Shipping Companies Merchant Vessel Crew USPACOM Local Navies	No
Prevent ships from being captured by pirates	Shipping Companies Merchant Vessel Crew Shipping Company Customers USPACOM Local Navies	Yes
Protect Crew	Shipping Companies Merchant Vessel Crew	Yes
Deny Hostile Access to ship	Shipping Companies Merchant Vessel Crew Shipping Company Customers	Yes
Halt Maritime piracy worldwide	Shipping Companies Merchant Vessel Crew Shipping Company Customers USPACOM IMO Local Navies	No
High Operational Availability	Shipping Companies	Yes
Cost-effective, affordable maintenance	Shipping Companies	Yes
Minimal installation requirements	Shipping Companies	Yes
Interoperability across commercial ship types	Shipping Companies IMO	Yes
Ruggedized Equipment	Shipping Companies Merchant Vessel Crew	Yes
Ease of Use	Shipping Companies Merchant Vessel Crew	Yes
Cost-effective	Shipping Companies Shipping Company Customers	Yes
Low Operational and Support Costs	Shipping Companies Shipping Company Customers	Yes
Low logistical Impact	Shipping Companies Merchant Vessel Crew Shipping Company Customers	Yes
Effective against all pirate vessels	Shipping Companies Merchant Vessel Crew USPACOM IMO Local Navies	Yes

3. Effective Need

The analysis of stakeholder needs showed that preventing pirate control of a merchant vessel is paramount. The prevention of pirate control however, is in every case precluded by the pirates themselves boarding the vessel. Therefore, the system solution must economically and efficiently prevent pirates from boarding commercial vessels in Indonesian waters, maintaining compliance with international maritime law. This anti-boarding system concept was given the name "Boarding Prevention System" or BPS. A context diagram detailing the boundaries of the system and interaction with external entities is shown below in Figure 9. The context diagram serves as a pictorial representation of the overall flow of information and material. Four primary external entities are listed: the commercial vessel, friendly forces, a fishing fleet composed of pirates and neutral boats, and overtly hostile pirate vessels.

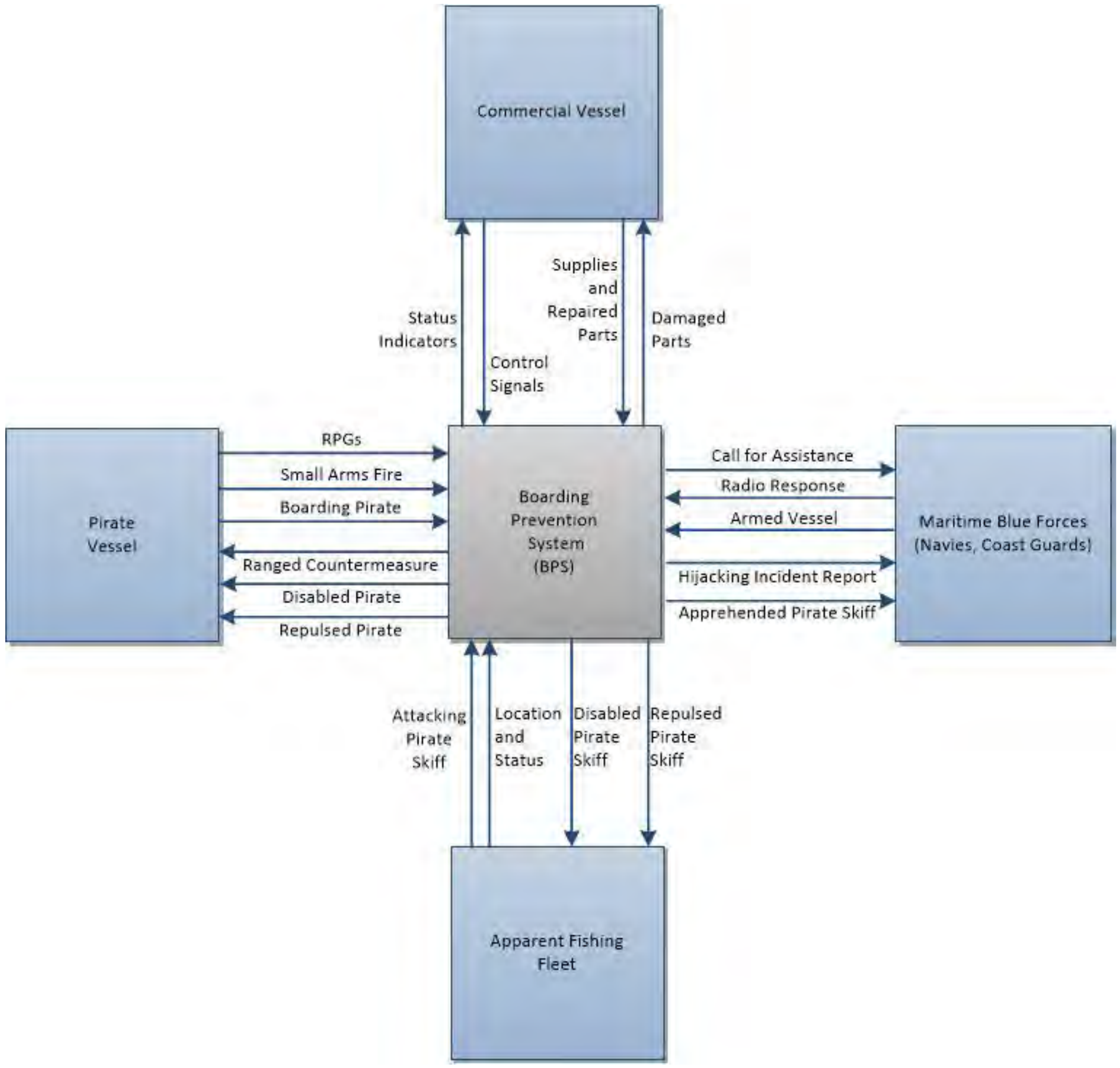


Figure 9. Context Diagram

This figure depicts the boundaries of the BPS and how the system interacts with external entities.

B. REQUIREMENTS

1. Concept of Operations (CONOPS)

The top-level CONOPS includes a commercial shipping vessel, pirate skiffs, and rescue forces. Pirate vessels approach the commercial vessel during transit and attempt to hijack the ship. The commercial vessel will immediately call for aid, and attempt to hold off the attacking forces until friendly vessels arrive. Once friendly naval or coast guard forces approach, the pirates will scatter in the face of superior firepower. Figure 10 illustrates the CONOPS, and is overlaid over a map of Indonesia.

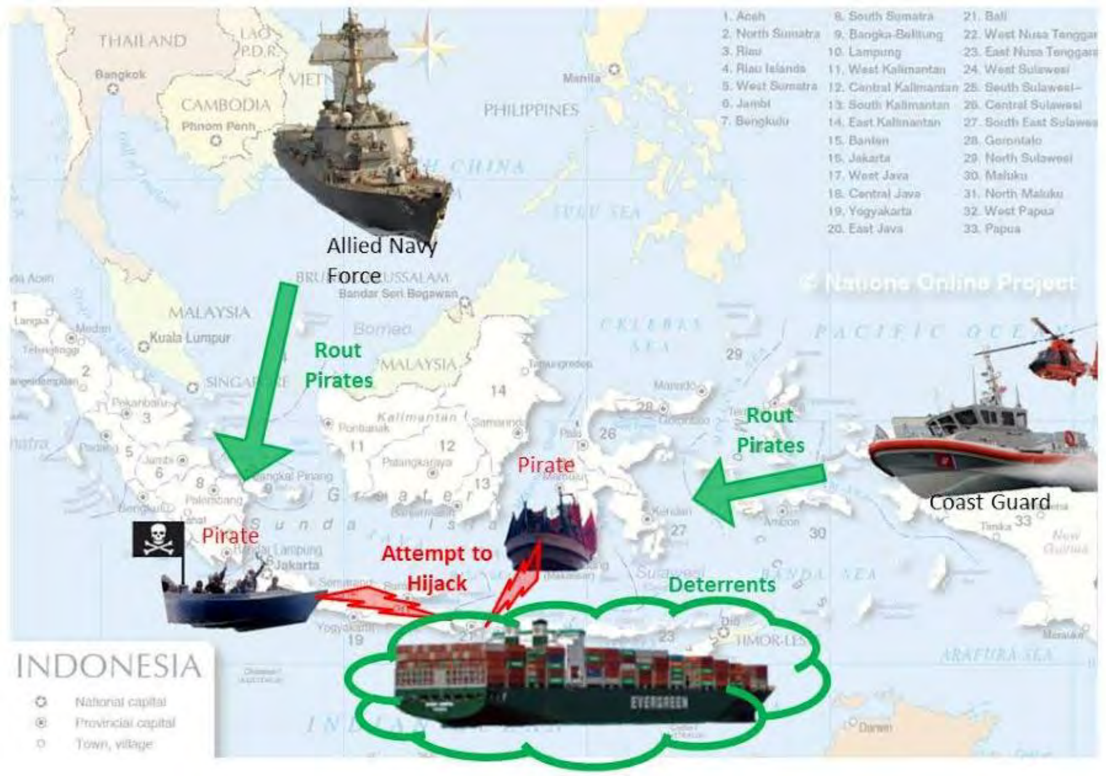


Figure 10. Top-Level CONOPS

This figure depicts the top-level operation of system. Pirates attack the commercial ship, which defends using onboard countermeasures until friendly military or coast guard arrive to rout pirates.

The CONOPS aided the effort of further defining the situation in which the BPS would be used by focusing research on the maximum time required for the BPS to operate before assistance would arrive. Once this was determined, efforts were focused on analyzing the boarding attempt, generating an OV-1 diagram which divides the operational theater into four ranges, from detection point to interception. Defensive systems will focus on either one or multiple ranges, and will use some combination of active and passive countermeasures to prevent hijacking. The OV-1 is shown in Figure 11.

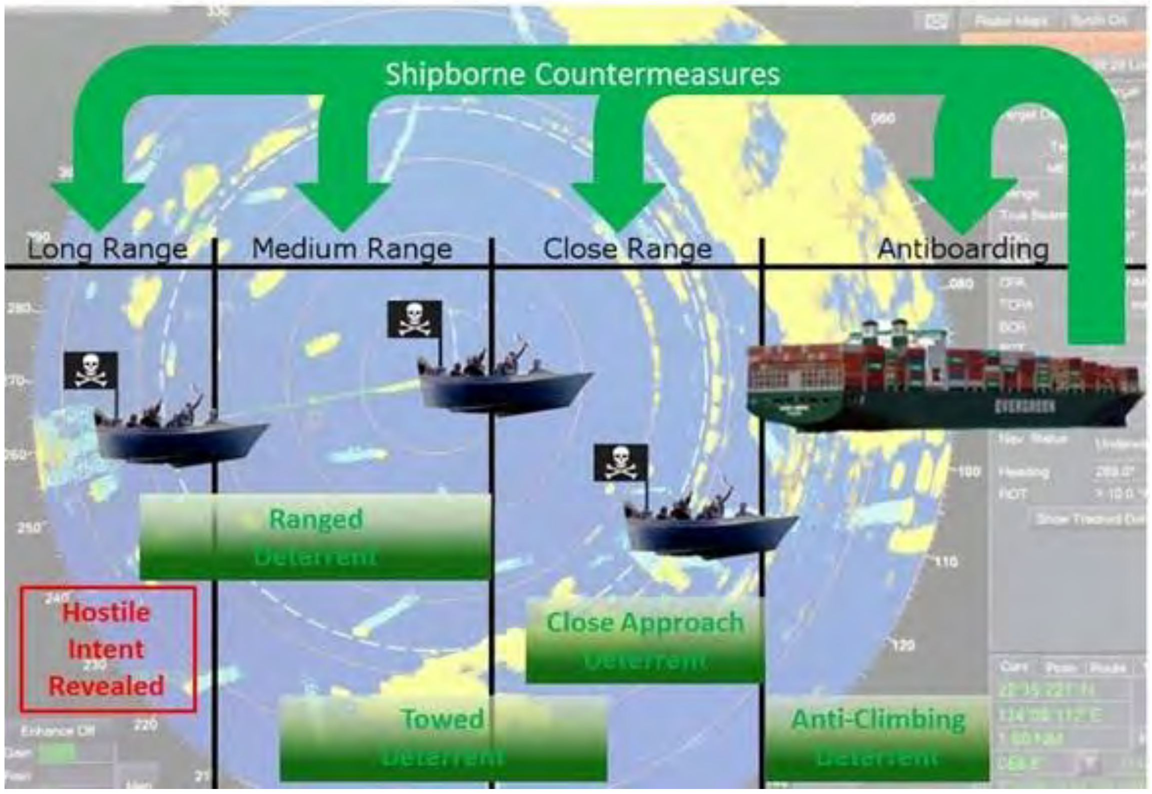


Figure 11. OV-1 Operational Concept

This figure depicts the use of ship-borne countermeasures against pirates at various ranges. Close through long ranged countermeasures act against skiffs, while anti-climbing countermeasures act only on pirates attempting to scale the ship.

2. Design Reference Mission

A Design Reference Mission was constructed to define a baseline threat scenario. The overall scenario consists of commercial vessels traveling through pirate infested travel routes surrounding Indonesia. The operational environment, shown in Figure 12, consists of a cargo ship attacked by pirate skiffs 100 nm from an allied military base. Aid is requested, and the military ship launches immediately on an intercept course. Pirates attempt to board the commercial ship and obtain control of it; the ship deploys the anti-boarding defensive system to prevent the loss of the ship.

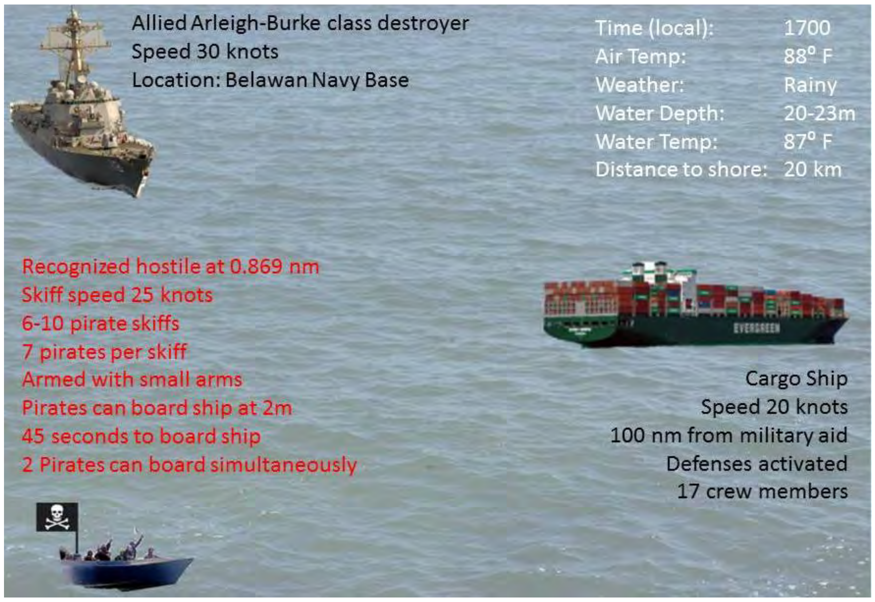


Figure 12. Operational Environment

This figure depicts the operational environment and the key entities involved in a typical scenario.

The scenario assumes that the pirate attack will cease if driven off or friendly vessels arrive to aid the commercial ship. The mission is defined as a success if the pirates are prevented from gaining control of the commercial ship. Success scoring is based on the following factors:

- Pirates are prevented from boarding ship through various methods
- Ship maneuvers to safe waters OR Navy task force arrives
- Loss of crew life is minimized

3. Functional Requirements

The top-level requirement of the system is the prevention of pirates from boarding the commercial vessel and gaining control of it. A decomposition of this requirement is shown below in Figure 13. The top-level requirement can be accomplished by impeding the entry routes of the pirates, forcing the pirate skiffs away from the vessel, or by degrading pirate capabilities. Increasing the distance between pirate skiffs and the commercial vessel can be accomplished in two ways: by increasing operational range or by a system configuration that decreases the pirate craft's ability to remain in the operational range. Finally, the pirate capabilities can be degraded by reducing pirate craft maneuverability, neutralizing pirate crew, or impairing pirate communications.

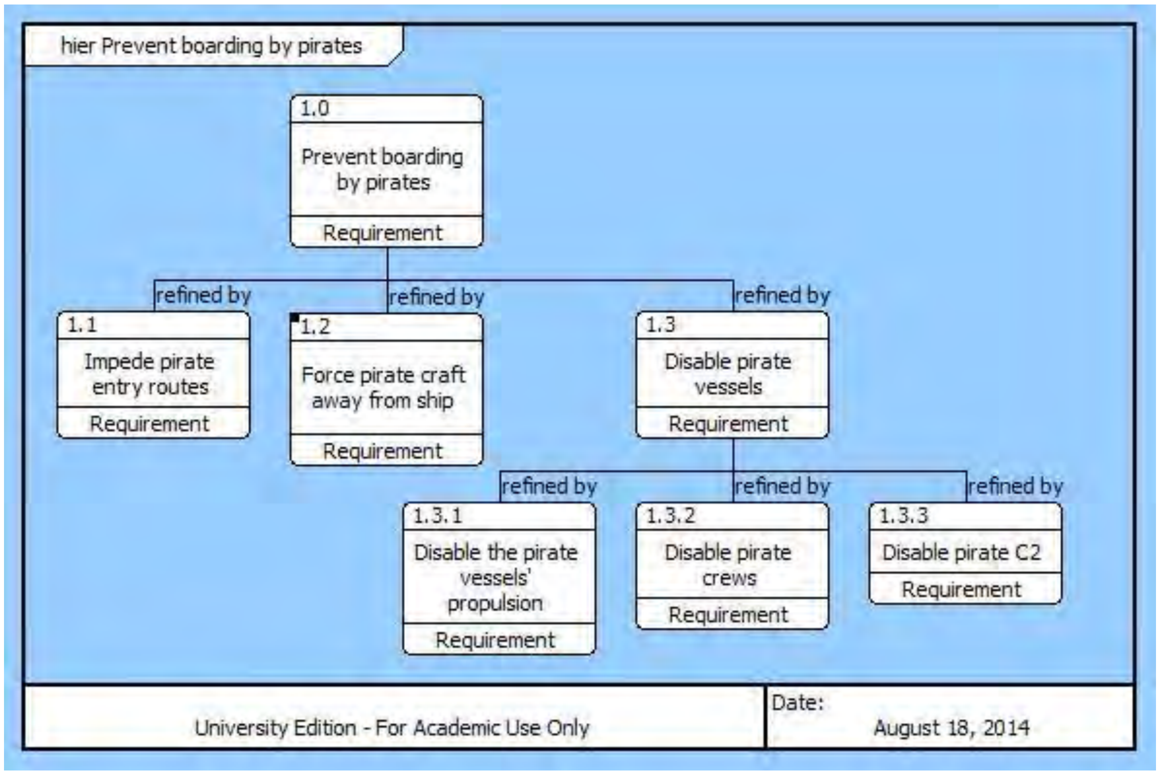


Figure 13. Piracy Prevention System Requirements

This figure depicts the requirements for the BPS, showing the breakdown of the top-level function into more specific system requirements.

The requirements generated from the decomposition of the top-level requirement are described in Table 5.

Table 5: Functional Requirements

This table identifies the functional requirements from the stakeholder needs and provides additional details.

#	**Functional Requirement Name**	**Requirement Detail**
1.0	Prevent boarding by pirates	the system shall prevent a pirate boarding long enough for help to arrive.
1.1	Impede pirate entry routes	the system shall impede pirate boarders from using their typical boarding routes.
1.2	Force pirate craft away from ship	the system shall push pirate vessels away from the ship to a distance of greater than 3 meters.
1.3	Disable pirate vessels	the system shall degrade the pirates' ability to execute their mission by disabling propulsion, crew capability, and ability to coordinate.
1.3.1	Disable the pirate vessels' propulsion	the system shall disable the pirate crafts' ability to keep up with the commercial vessel.
1.3.2	Disable pirate crews	the system shall neutralize pirate crew and boarders as participants in the attack using nonlethal methods
1.3.3	Disable pirate Command and Control (C2)	the system shall prevent the pirates from effectively coordinating their assault

4. Non-Functional Requirements

The non-functional requirements listed in Table 6 represent the stakeholder needs that were unmet by the functional requirements. These requirements were given consideration during the trade study of COTS countermeasures in the System Architecture phase.

Table 6. Non-Functional Requirements

This table identified the functional requirements from the stakeholder needs and provides additional details.

Non-functional Requirement Name	**Requirement Detail**
Compliance with international law	the system shall comply with all applicable laws for international waters
High system reliability	the system shall incorporate a MTBF sufficient for the vessel to complete 10 voyages between average failures.
Cost-effectiveness	the system shall have an affordable purchase cost and minimize Operations & Support costs
Standard deck equipment interface and installation	the system shall require minimal installation time and shall not require a dry dock or non-standard tools
Interoperability	the system shall be usable on all commercial ships manufactured globally in the last 30 years
Affordable and simple logistics support	the system shall not require unique or hard-to-obtain consumables nor shall it require replacement of routine consumables more frequently than once per quarter
Increase the range at which pirate craft can remain near the ship unmolested	the system shall increase the range pirate craft must remain at in order to avoid countermeasures.
the system shall decrease the pirate crafts' ability to remain within the protected region	the system shall reduce the time that a pirate craft will remain within the vicinity of the vessel protected by the system.
Low environmental impact	the system shall have minimal impact on the environment
	the system shall not produce any waste harmful to the crew or requiring enhanced disposal techniques
Low maintenance	the system shall not require active oversight for deployed passive systems
	the system shall not require more scheduled maintenance than standard deck equipment
Built-in, automatic notification of failures	the system shall create an audio-visual notification if a failure occurs that would prevent mission success

C. FUNCTIONAL ARCHITECTURE

After analyzing the requirements for the BPS, the functional development process generated an overall function for the system, which was then decomposed into more defined functional elements. The overall function was for the system to prevent pirates from boarding the commercial vessel, and the decomposed functions providing the specific tactical functions (second and third level functions) can be found in Figure 14. Several identified requirements were also determined to be non-viable for implementation in the BPS. Specifically, neutralizing pirate crew via lethal measures will encounter numerous ethical, legal, and safety guidelines. Additionally, the impairment of communications is not a viable approach; pirates typically do not coordinate between skiffs once an attack is started. Therefore, the project shall prevent pirates from boarding the commercial vessel by some combination of impeding entry routes, forcing pirate craft away from the target ship, or degrading pirate capabilities.

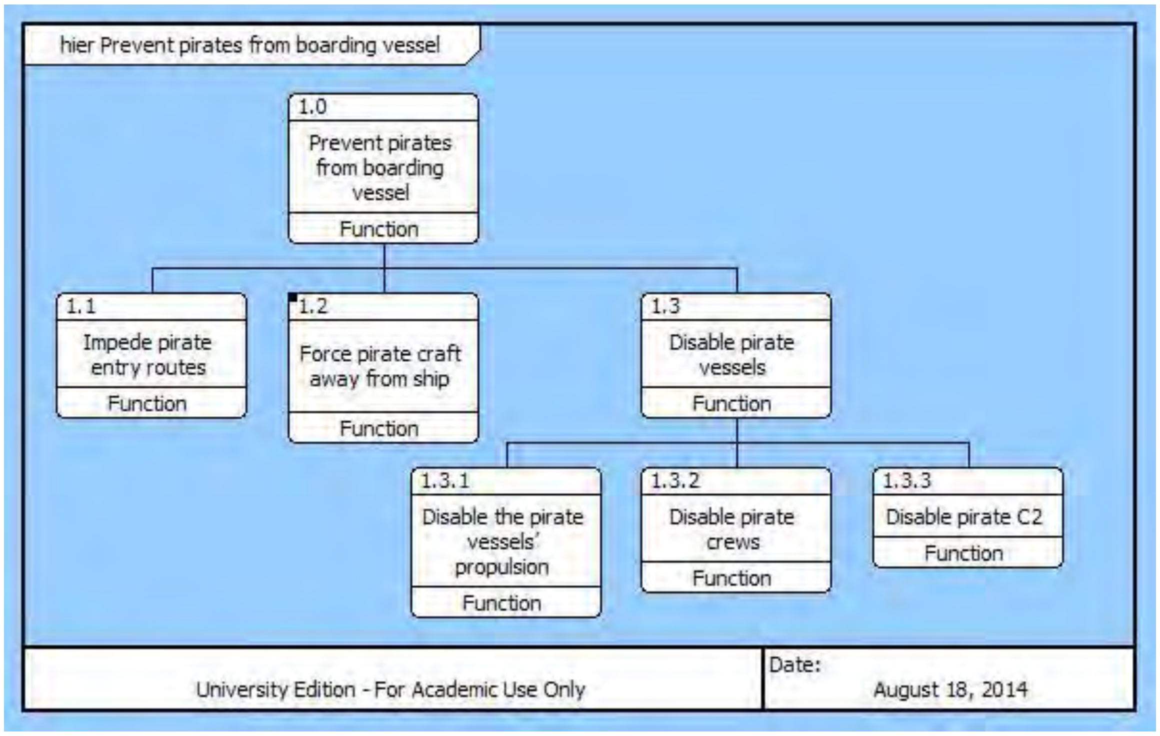

hier Prevent pirates from boarding vessel
1.0
Prevent pirates from boarding vessel
Function
1.1
Impede pirate entry routes
Function
1.2
Force pirate craft away from ship
Function
1.3
Disable pirate vessels
Function
1.3.1
Disable the pirate vessels' propulsion
Function
1.3.2
Disable pirate crews
Function
1.3.3
Disable pirate C2
Function
University Edition - For Academic Use Only
Date:
August 18, 2014

Figure 14. Top-level Functional Hierarchy of BPS

This figure depicts the functional decomposition from the top-level function of boarding prevention to lower-level functions detailing how the overall function is accomplished. This diagram covers the general functions performed by the countermeasures.

Table 7. BPS Functions

This table identifies the system functions from the functional requirements and provides additional details.

#	Function Name	Function Detail
1.0	Prevent boarding by pirates	the system shall prevent a pirate boarding long enough for help to arrive through use of countermeasures.
1.1	Impede pirate entry routes	Block routes of ingress to the commercial vessel or increase difficulty and time required to board by using obstacles and hoses.
1.2	Force pirate craft away from ship	Physically push pirate craft out of boarding range or create zones pirate craft will avoid.
1.3	Disable pirate vessels	Degrade the pirates' ability to execute their mission by disabling propulsion, crew capability, and ability to coordinate by using countermeasures.
1.3.1	Disable the pirate vessels' propulsion	Foul pirate craft propellers with projectiles, lines, and nets.
1.3.2	Disable pirate crews	Non-lethally subdue pirates or knock them into the water.
1.3.3	Disable pirate C2	Prevent the pirates from effectively coordinating their assault by using distraction systems.

D. SYSTEM ARCHITECTURE

1. Viable Solutions

The output from the development of the system architecture provided a list of 25 possible piracy countermeasures listed below:

Razor Wire

Razor wire is a low-cost and widely used method of creating a defensive barrier. It is typically comprised of metal strips with sharp edges or barbs placed throughout its length. The principle method of deterrence is by posing a high risk of lacerations to the trespasser. often times the metal strips are laid in a spiral and placed under tension so that if they are cut, the wire snips and strikes the trespasser.

Electrified Wire

This defensive barrier is razor wire that has an electrical source provide a pulsed high voltage through the wire, potentially stunning an intruder making contact with the wire.

Pepper Spray

A crew operated aerosol system mounted on the side of a ship, can release up to 300 gallons of pepper spray onto individuals boarding the ship.

Long Range Acoustic Device (LRAD)

"The Long range acoustic device is a non-lethal anti-piracy device that uses a pain inducing sound beam to drive away pirates. The sonic weapon produces high-pitched noise that is higher than the tolerance level of an average human being. LRAD has been used on few cargo and cruise ships until now" ("18 Anti-Piracy Weapons" 2013). Range: 10 to 3,000 meters. (http://www.lradx.com/site/content/view/323)

Anti-Piracy Laser Device (non-lethal)

"The anti-piracy laser device uses non-lethal laser beam to provide a visual warning to pirates and distract them temporarily. The laser device can be used during both day and night, and can be easily operated by the ship's crew" ("18 Anti-Piracy Weapons" 2013). Developed by BAE, tests suggest it is effective at a distance of 1.5km.

Nets-Boat Traps

"Boat trap is a type of ballistic net which can be used to stop pirates' boats when they come near to a merchant ship. When in water, the net ensnares the propellers of the boats that disable the vessel, preventing it from moving forward" ("18 Anti-Piracy Weapons" 2013).

Slippery Foam

"Slippery foam or Anti-traction material is a non-lethal substance that can be used to make the deck or sides of a ship slippery to avoid pirates from climbing it. The highly viscous substance substantially reduces traction of anything that comes in contact with it, making it difficult to walk or stand" ("18 Anti-Piracy Weapons" 2013).

Foul-Smelling Liquid (Liquid Countermeasure System)

"An anti-piracy technology by the International Maritime Security Network of U.S. involves showering approaching pirates with slick, foul-smelling green liquid, which stinks and burns. The burning sensation and the nasty stink caused by the liquid forces pirates to jump into the water, thus stopping a possible pirate attack" ("18 Anti-Piracy Weapons" 2013).

Anti-Boarding device (Razor Wire Canister)

"Anti-boarding device is an anti-piracy method which uses canisters with sharp razor wires to prevent pirates from boarding the ship. The wires act as a barrier between the pirates and the ship, which thwarts forward movement of pirates" ("18 Anti-Piracy Weapons" 2013).

Compressed Air - Ship Bourne Shore Launcher

"the Ship Bourne Shore Launcher is a product of a UK based company. The Buccaneer Ship Bourne Shore Launcher is a cannon shaped device which uses compressed air to fire a variety of projectiles. The power and lethality of the projectiles used can vary according to the distance of the pirates from the ship" ("18 Anti-Piracy Weapons" 2013)

P-trap Anti-Piracy Fouling lines

"[The] P-trap concept is a non-lethal system which helps prevent pirates from boarding ships. The system carries thin lines which float at the water level around the sides of the vessel. When pirate skiffs/boats come in contact with the lines, the later gets entangled with the engine and disable the vessel" ("18 Anti-Piracy Weapons" 2013). Effective if run into P-trap, limited range, limited lines. According to FAQs, after four attacks, nine P-trap lines were used, there are 20 traps per side, and so multiple waves of craft can defeat P-trap.

Anti-Piracy Curtain

"Designed by a division of Japan's NYK group along with hose manufacturer Yokoi, the anti-piracy curtain is a unique method to keep pirates from climbing the ships. The system consists of a series of hoses which are dangled on the port and starboard sides of the vessel. Sea water is passed through the nozzles at a force of 0.2 Mega Pascal, which makes the hoses go in unpredictable whirling motion, generating enough force to seriously hurt anyone who gets in the way" ("18 Anti-Piracy Weapons" 2013).

Non-lethal/Stun Grenade

"Stun grenade or flash grenade is a non-lethal anti-piracy device which produces a blinding flash of light and loud noise. Stun grenades are used to temporary disorient a pirate's senses without causing any kind of permanent injury" ("18 Anti-Piracy Weapons" 2013).

Dazzle Gun

"Dazzle guns is a type of laser weapon which uses green light to disorient and temporary blind the pirates. The concentrated blast of green light can be used during both day and night" ("18 Anti-Piracy Weapons" 2013).

Rubber Ball Grenade

"Rubber ball grenade as a non-lethal weapon sprays rubber bullets on detonation. The anti-piracy grenade also produces light and sound which can be used to deter pirates from coming towards the ship" ("18 Anti-Piracy Weapons" 2013).

Active Denial System - Pain Ray

"Officially known as the Active Denial System (ADS), the Pain Ray is a non-lethal weapon which transmits a narrow beam of electromagnetic energy to heat the skin without causing permanent damage. The wave penetrates beneath the skin which causes unbearable burning sensation, forcing pirates to run away or jump overboard" ("18 Anti-Piracy Weapons" 2013). Operational range and specific time it takes to achieve an effect is access restricted.

Anti-Piracy Fire Hoses

"Ship's fire hoses or special Anti-piracy fire hoses are often used to fight pirates trying to board the ships. These high pressure water hoses are extremely powerful and effective to fight pirates. Special anti-piracy fire hoses also come with semi-automatic and remote control system" ("18 Anti-Piracy Weapons" 2013).

Laser Weapon Systems (LaWS) (lethal)

LaWS is a system based on a design developed by the Navy Research Lab and engineers at the Naval Sea Systems Command and Naval Surface Warfare Center Dahlgren. Its purpose is not to vaporize enemy ships but to provide a low-cost way for the Navy to defend against drones, small boats, light aircraft, and missiles at ranges of about a mile.

Hire Security Personnel to train self-defense/combat

This countermeasure entails hiring private military contractors to provide advanced self-defense techniques and drilling the crew of commercial vessels in the use of small arms for self defense

Agree to Ransom Conditions of Pirates

This "countermeasure" is included as a baseline to measure other countermeasure options against as this is the status-quo solution being used currently.

Hire Security Team on Board

Hiring private military contractors to provide security for commercial vessels

Weapons

This countermeasure entails providing the crew with small arms and training the crew in basic safety measures.

Smaller Vessels w/ less cargo, therefore lower value target

This solution would involve changing the composition of shipping fleets to emphasize smaller vessels that present less lucrative targets for pirates.

Change Ship Route

Travel via a safer but less-efficient route.

Anti-piracy guardrails

Plastic/rubber matrix formed over the guardrails of the ship similar to bumpers for bowling lanes. Simply put these on after getting into open water, then remove before docking. If the boat's above board height is >8 ft., then no reported instances of failure. This will not allow grappling hooks to come onboard and hook onto anything. Developed for British merchant ships.

2. Trade Study

A trade study was conducted to reduce the available countermeasures via a decision matrix, found in Table 8. The decision matrix scores were based on the convention that a rating of one was desired and a rating of five was undesirable. Each factor was ranked according to an overall score determined by dividing individual countermeasure scores by the theoretical maximum. The top five countermeasures were selected for modeling to determine success in preventing pirate attacks.

Table 8. Decision Matrix Analysis of Potential Countermeasures

This table shows the decision matrix of 25 possible countermeasures. Each countermeasure was ranked according to generated MOPs and assigned a normalized overall score based on the countermeasure score divided by the theoretical maximum.

	Deployment	Ease of Use	Scheduled Maintenance	Cost	Resistance to Attack	Required Logistics	Effect on Crew	
Multiplier	0.1304348	0.1304348	0.1304348	0.2173913	0.1304348	0.0434783	0.2173913	
Countermeasure								**Score**
Razor Wire	1	2	1	1	2	1	3	86.09%
Electrified Wire	2	2	3	3	4	2	4	59.13%
Pepper Spray	1	2	1	1	2	3	3	84.35%
Long Range Acoustic Device (LRAD)	3	3	4	5	4	3	2	50.43%
Anti-Piracy Laser Device (non-lethal)	4	3	4	5	5	3	1	49.57%
Nets-Boat Traps	5	3	3	3	3	3	1	63.48%
Slippery Foam	2	2	3	3	2	4	1	75.65%
Foul Smelling Liquid - Liquid Countermeasure System	3	2	3	3	2	4	3	64.35%
Anti-Boarding device - Razor Wire Canister	2	2	2	3	4	2	1	74.78%
Compressed Air - Ship Bourne Shore Launcher	3	4	2	2	2	3	2	71.30%
P-trap Anti-Piracy Fouling lines	2	1	2	2	4	2	1	81.74%
Anti-Piracy Curtain	2	2	2	1	3	2	2	81.74%
Non-lethal/Stun Grenade	2	1	1	2	4	2	4	71.30%

Dazzle Gun	3	2	3	3	3	2	1	72.17%
Rubber Ball Grenade	4	2	2	2	3	3	3	66.96%
Active Denial System - Pain Ray	3	4	3	5	2	3	1	60.00%
Anti-Piracy Fire Hoses	2	2	2	2	2	2	2	80.00%
Laser Weapon Systems (LaWS) (lethal)	4	3	5	5	2	3	3	46.09%
Hire Security Personnel to train crew in self-defense/combat	1	3	2	5	2	4	2	65.22%
Agree to Ransom Conditions of Pirates	2	2	1	5	3	3	5	53.04%
Hire Security Team on Board	2	2	2	5	2	3	3	61.74%
Weapons	2	1	3	3	3	3	5	59.13%
Smaller Vessels w/ less payload, therefore lower value target	1	3	1	5	4	4	3	58.26%
Change Ship Route	2	2	1	2	5	4	2	73.04%
Anti-piracy guardrails	3	1	2	3	2	3	1	79.13%

3. Selected Countermeasures

Table 9 shows the five piracy countermeasures that were incorporated into the model based on their scores in their respective categories in the decision matrix. The two passive defense countermeasures are razor wire and P-Traps. For active defense, the Anti-Piracy Curtain and Anti-Piracy Fire Hoses scored the highest. The active offense countermeasure is the compressed air cannon.

Table 9. Selected Countermeasures

This table shows the top five countermeasures from the trade study, which are selected for modeling.

Countermeasure	Category	Score
Razor Wire	Passive Defense	0.86
P-Trap Anti-Piracy	Passive Defense	0.82
Anti-Piracy Curtain	Active Defense	0.82
Anti-Piracy Fire Hoses	Active Defense	0.80
Compressed Air Cannon	Active offense	0.71

Razor wire is a low cost and widely used method of creating a defensive barrier. It is typically comprised of metal strips with sharp edges or barbs placed throughout its length. The principle method of deterrence is by posing a high risk of lacerations to the trespasser. often times the metal strips are laid in a spiral and placed under tension so that if they are cut, the wire whips and strikes the trespasser. It is designated as a passive defense system because once it is set, the system is turned “on.”

P-Traps are a passive defensive system primarily distributed by Westmark BV. This anti-piracy system is comprised of heavy lines hung of booms mounted on the sides of a vessel. It is a close range system as their effectiveness is only as far as the boom extends which is currently maxed out at 12 meters. Booms can be mounted so that the lines trail from bow to stern on both the port and starboard sides, and additional booms can be mounted straight off the stern to provide a rear defense. The primary deterrence mechanism is disabling an attacking vessel's propeller by entangling the heavy lines onto them degrading their maneuverability considerably.

The anti-piracy Curtain is a countermeasure that uses intense water pressure contained in heavy gauge hoses to thwart would be boarders to a vessel. Hung off the sides of the vessel, the intense pressure causes the hoses to whip wildly along the flanks of the ship. A would-be boarder who decides to climb a vessel equipped with the curtain would be subject to blunt force trauma from the turbulent hoses. It was considered an active defense because the system would have to be engaged prior to a pirate attack and turned off after the attack.

Anti-piracy Fire Hoses are modelled on UNIFIRE's anti-pirate water cannons system. Typical installations on commercial vessels include six water cannons, three for the port and starboard side each. These cannons can be remote operated or automated through integration with a radar system. They fire up to 50 liters of water per minute, at a pressure of 10 bars, and with an effective range of up to 90 meters. These hoses can be used to keep pirates at bay by using the pressure exerted to keep them outside a critical range, or, against smaller skiffs, they can be used to quickly fill and sink the attacker's vessel. These are also active defense systems because they must also be engaged prior to the pirate attack.

The Compressed Air Cannon is a pneumatic projectile launcher that can be loaded with a wide range of munitions. For the anti-piracy application, the non-lethal projectiles were selected. The launchers fire net-like non-lethal projectiles intended to entangle with the attacking skiff's propellers rendering them immobile or severely handicapped with respect to maneuverability. Their reported effective range is up to 300 meters.

E. MODELING AND SIMULATION RESULTS

1. Modeled System Configurations

The full list of selected system configurations can be found in Appendix A. The system configurations selected to be simulated were each run 100 times, validated, and then run 1000 times.

2. Results

For each of the twenty-four system configurations, the DRM was simulated 1000 times, and the total number of successful runs divided by the number of total runs to average the success rate. Four DRM runs contained all mission failures, three DRM runs contained all mission successes, and the remainder fell between 0.8 and 0.99. The survival percentages for each configuration are listed below in Table 10.

Table 10. Simulation Results

This table shows which countermeasures are enabled for each simulation run, and what survival rate each system configuration obtained per set of 100 and 1000 runs, respectively.

	Passive Defense		**Active Defense**		**Active offense**	**Survival MOE Results**	
Run #	**Razor Wire**	**Pirate Curtain**	**P-Trap**	**Water Cannon**	**Compressed Air Cannon**	**100x Runs**	**1000x Runs**
1	off	off	On	off	off	0	0.000
2	On	off	On	off	On	0.99	0.972
3	On	On	On	off	On	0.98	0.996
4	On	On	On	On	off	0.99	0.997
5	On	off	On	On	off	0.92	0.919
6	off	off	On	On	On	0.9	0.864
7	On	On	On	off	off	0.99	0.994
8	On	off	off	On	off	0.87	0.89
9	off	off	On	On	off	0	0.000
10	On	On	On	On	On	1	0.996
11	On	On	off	off	off	0.99	0.994
12	On	off	On	On	On	0.99	0.984
13	On	On	off	On	off	1	0.997

14	On	off	off	off	off	0.87	0.85
15	off	On	On	On	On	0.99	0.995
16	On	off	On	off	off	0.8	0.085
17	On	On	off	On	On	0.98	0.997
18	On	On	off	off	On	1	0.997
19	off	off	On	off	On	0.92	0.847
20	off	On	On	off	off	0	0.000
21	On	off	off	off	On	0.94	0.973
22	On	off	off	On	On	0.99	0.981
23	off	On	On	On	off	0	0.000
24	off	On	On	off	On	0.98	0.993

3. Model Limitations

Several assumptions and limitations are identified for the model as it exists today. Appendix D lists the areas where it is identified that the model relies upon assumptions due to a lack of data available today, or where the data sources are considered less than unquestionable. Appendix E lists known areas where the model has significant room for improvement and could be focused on in any follow-on effort.

F. COST ANALYSIS RESULTS

Each set of simulation runs evaluated a specific system configuration. The five-year Net Present Value (NPV) system cost for each system configuration was determined using the formulas discussed in Section II and the results are listed in Tables 11 through 16. Not all cost data was obtainable from the countermeasure manufacturers as some systems, such as the Compressed Air Cannon, had not been integrated with a commercial shipping vessel. A best estimate was made for integration costs and other areas without hard data.

Table 11. System Configuration Cost

This table shows the total five-year NPV cost of all installed countermeasures for a given configuration.

System Configuration	**Razor Wire**	**P-Trap**	**Curtain**	**Fire Hose**	**Compressed Air Cannon**	**Sum**
1	$394,889	$0	$0	$0	$0	$394,889
2	$394,889	$361,871	$0	$0	$802,610	$1,559,370
3	$394,889	$361,871	$0	$978,831	$802,610	$2,538,200
4	$394,889	$361,871	$235,280	$978,831	$0	$1,970,871
5	$394,889	$361,871	$235,280	$0	$0	$992,040
6	$394,889	$0	$235,280	$0	$802,610	$1,432,779
7	$394,889	$361,871	$0	$978,831	$0	$1,735,591
8	$0	$361,871	$235,280	$0	$0	$597,151
9	$394,889	$0	$235,280	$0	$0	$630,169
10	$394,889	$361,871	$235,280	$978,831	$802,610	$2,773,480
11	$0	$361,871	$0	$978,831	$0	$1,340,701
12	$394,889	$361,871	$235,280	$0	$802,610	$1,794,650
13	$0	$361,871	$235,280	$978,831	$0	$1,575,981
14	$0	$361,871	$0	$0	$0	$361,871
15	$394,889	$0	$235,280	$978,831	$802,610	$2,411,610
16	$394,889	$361,871	$0	$0	$0	$756,760
17	$0	$361,871	$235,280	$978,831	$802,610	$2,378,591
18	$0	$361,871	$0	$978,831	$802,610	$2,143,311
19	$394,889	$0	$0	$0	$802,610	$1,197,499
20	$394,889	$0	$0	$978,831	$0	$1,373,720
21	$0	$361,871	$0	$0	$802,610	$1,164,480
22	$0	$361,871	$235,280	$0	$802,610	$1,399,760
23	$394,889	$0	$235,280	$978,831	$0	$1,609,000
24	$394,889	$0	$0	$978,831	$802,610	$2,176,330

Table 12. Water Cannon Total Ownership Cost

This table shows a five-year NPV for the Water Cannon

NPV Water Cannon	**Year**					
Item	0	1	2	3	4	5
System Purchase	$704,000	$0	$0	$0	$0	$0
System Integration	$17,600	$0	$0	$0	$0	$17,600
Consumable Purchase	$1,000	$2,000	$2,000	$4,000	$4,000	$6,000
Routine Maintenance	$0	$7,680	$7,680	$15,360	$15,360	$23,040
Unscheduled Maintenance	$0	$5,760	$5,760	$11,520	$11,520	$17,280
IT Support	$0	$0	$0	$0	$0	$0
Documentation	$20,704	$10,352	$10,352	$10,352	$10,352	$10,352
Training	$7,135	$0	$0	$0	$0	$0
Inflation multiplier Estimate	1	1.06	1.12	1.19	1.26	1.34
Subtotal(Current Value)	$750,440	$27,339	$28,980	$49,108	$52,054	$99,393
Interest Multiplier Estimate	1	1.03	1.07	1.1	1.14	1.18
total (Present Value)	$750,440	$26,466	$27,158	$44,550	$45,715	$84,499
Net Present Value						$978,830

Table 13. Compressed Air Launcher total Ownership Cost

This table shows a five-year NPV for the Compressed Air Launcher

NPV Compressed Air Launcher	**Year**					
Item	0	1	2	3	4	5
System Purchase	$372,900	$0	$0	$0	$0	$0
System Integration	$211,200	$0	$0	$0	$0	$0
Consumable Purchase	$5,000	$0	$5,000	$0	$10,000	$0
Routine Maintenance	$0	$2,560	$2,560	$5,120	$5,120	$7,680
Unscheduled Maintenance	$0	$8,000	$8,000	$16,000	$16,000	$32,000
IT Support	$1	$1	$2	$3	$4	$5
Documentation	$20,705	$10,352	$10,352	$10,352	$10,352	$10,352
Training	$7,135	$0	$0	$0	$0	$0
Inflation multiplier Estimate	1	1.06	1.12	1.19	1.26	1.34
Subtotal(Current Value)	$616,941	$22,168	$29,117	$37,488	$52,363	$66,961
Interest Multiplier Estimate	1	1.03	1.07	1.1	1.14	1.18
total (Present Value)	$616,941	$21,460	$27,287	$34,009	$45,986	$56,928
Net Present Value						$802,610

Table 14. P-Trap Total Ownership Cost

This table shows a five-year NPV for the P-Trap

NPV P-trap	Year					
Item	0	1	2	3	4	5
System Purchase	$62,400	$0	$0	$0	$0	$0
System Integration	$1,040	$0	$0	$0	$0	$0
Consumable Purchase	$3,000	$6,000	$6,000	$12,000	$12,000	$18,000
Routine Maintenance	$0	$7,680	$7,680	$15,360	$15,360	$23,040
Unscheduled Maintenance	$0	$7,760	$7,760	$15,520	$15,520	$23,280
IT Support	$0	$0	$0	$0	$0	$0
Documentation	$20,705	$10,352	$10,352	$10,352	$10,352	$10,352
Training	$7,135	$0	$0	$0	$0	$0
Inflation multiplier Estimate	1	1.06	1.12	1.19	1.26	1.34
Subtotal(Current Value)	$94,280	$33,700	$35,722	$63,401	$67,205	$99,929
Interest Multiplier Estimate	1	1.03	1.07	1.1	1.14	1.18
total (Present Value)	$94,280	$32,623	$33,476	$57,517	$59,020	$84,955
Net Present Value						$361,871

Table 15. Pirate Curtain Total Ownership Cost

This table shows a five-year NPV for the Pirate Curtain

NPV Pirate Curtain	Year					
Item	0	1	2	3	4	5
System Purchase	$88,200	$0	$0	$0	$0	$0
System Integration	$12,600	$0	$0	$0	$0	$0
Consumable Purchase	$1,000	$0	$1,000	$0	$2,000	$0
Routine Maintenance	$0	$2,560	$2,560	$5,120	$5,120	$7,680
Unscheduled Maintenance	$0	$1,920	$1,920	$3,840	$3,840	$7,680
IT Support	$0	$0	$0	$0	$0	$0
Documentation	$20,705	$10,352	$10,352	$10,352	$10,352	$10,352
Training	$7,135	$0	$0	$0	$0	$0
Inflation multiplier Estimate	1	1.06	1.12	1.19	1.26	1.34
Subtotal(Current Value)	$129,640	$15,722	$17,789	$23,001	$26,906	$34,409
Interest Multiplier Estimate	1	1.03	1.07	1.1	1.14	1.18
total (Present Value)	$129,640	$15,220	$16,671	$20,867	$23,630	$29,253
Net Present Value						$235,280

Table 16. Razor Wire Total Ownership Cost

This table shows a five-year NPV for the Razor Wire

NPV Razor Wire	Year					
Item	0	1	2	3	4	5
System Purchase	$35,700	$35,700	$0	$0	$0	$0
System Integration	$4,200	$4,200	$4,200	$8,400	$8,400	$12,600
Consumable Purchase	$2,000	$4,000	$4,000	$8,000	$8,000	$12,000
Routine Maintenance	$0	$7,680	$7,680	$15,360	$15,360	$23,040
Unscheduled Maintenance	$0	$7,680	$7,680	$15,360	$15,360	$23,040
IT Support	$0	$0	$0	$0	$0	$0
Documentation	$20,705	$10,352	$10,352	$10,352	$10,352	$10,352
Training	$7,135	$0	$0	$0	$0	$0
Inflation multiplier Estimate	1	1.06	1.12	1.19	1.26	1.34
Subtotal(Current Value)	$69,740	$73,789	$38,104	$68,451	$72,558	$108,440
Interest Multiplier Estimate	1	1.03	1.07	1.1	1.14	1.18
total (Present Value)	$69,740	$71,432	$35,708	$62,098	$63,721	$92,191
Net Present Value						$394,889

G. SYSTEM CONFIGURATION SCORING

The modeled system configurations were plotted using the survival data from the modeling phase and the five-year cost from the cost analysis and can be found in Figure 15; the green circles (system configurations 21, 11, and 13) represent the "knee of the curve" and the non-dominated solutions that should be looked at closely in selecting a single system configuration for procurement. Table 17 contains the raw cost and survival percentages.

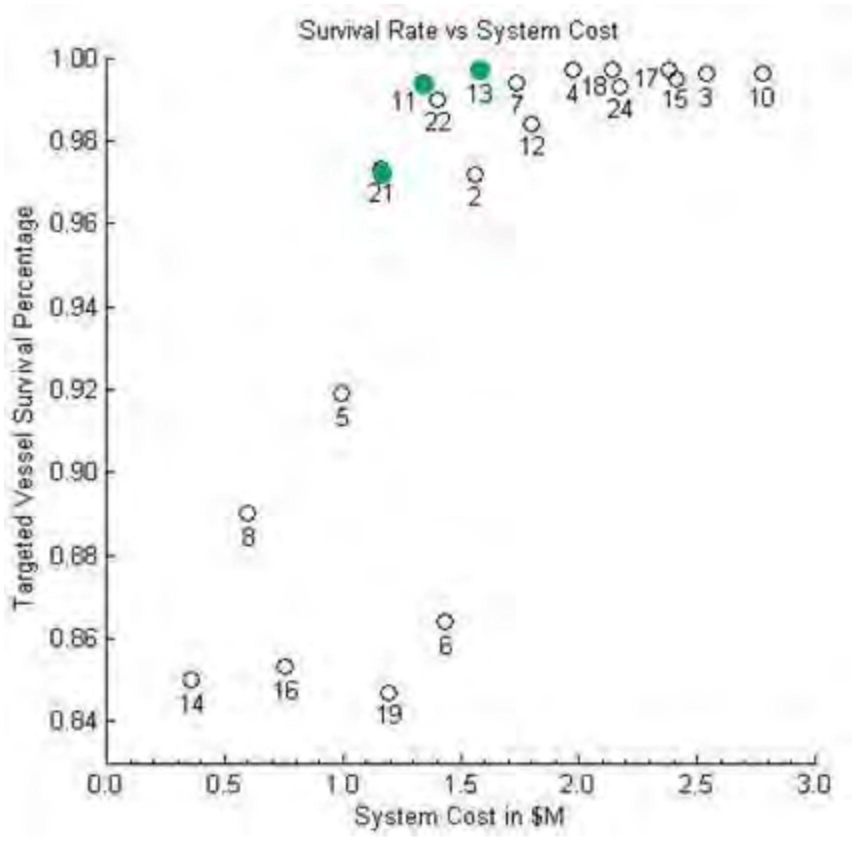


Figure 15. Cost vs. Survival Percentage

This figure depicts cost in $M vs survival percentage of each system configuration. The green circles (system configurations 21, 11, and 13) represent the "knee of the curve" and the non-dominated solutions that should be looked at closely in selecting a single system configuration for procurement.

Table 17. Cost and Survival Results

This table shows the system configurations and the associated cost and survival numbers. The green boxes (system configurations 11, 13, and 21) represent the "knee of the curve" and the non-dominated solutions that should be looked at closely in selecting a single system configuration for procurement.

Run #	1000x Survival MOE	5 Year Cost
1	0	$394,889
2	0.972	$1,559,370
3	0.996	$2,538,200
4	0.997	$1,970,871
5	0.919	$992,040
6	0.864	$1,432,779
7	0.994	$1,735,591
8	0.89	$597,151
9	0	$630,169
10	0.996	$2,773,480
11	0.994	$1,340,701
12	0.984	$1,794,650
13	0.997	$1,575,981
14	0.85	$361,871
15	0.995	$2,411,610
16	0.085	$756,760
17	0.997	$2,378,591
18	0.997	$2,143,311
19	0.847	$1,197,499
20	0	$1,373,720
21	0.973	$1,164,480
22	0.981	$1,399,760
23	0	$1,609,000
24	0.993	$2,176,330

THIS PAGE INTENTIONALLY LEFT BLANK

IV. CONCLUSIONS

Out of twenty-five available piracy countermeasures, five were selected by decision matrix. These five countermeasures were analyzed using a model simulating a commercial vessel defending against a pirate boarding attempt mimicking the conditions specified in a Design Reference Mission. Twenty-four unique combinations of the selected countermeasures were tested in the model.

After simulations were completed for all of the system configurations, the simulation results and cost estimates were imported into Microsoft Excel software to determine the effects of the countermeasure interactions. The five-year costs and effectiveness were ranked and combined into an overall ranking.

The results of the simulations and cost analyses showed three configurations that maximized cost-effectiveness. Usage of the P-Trap countermeasure combined with the Compressed Air Cannon provided a success rate of 97.3% with a five-year cost of \$1.164M/ship. A slightly more effective system configuration consists of the P-Trap countermeasure combined with the Fire Hose, with a success rate of 99.4% and a five year cost of \$1.341M/ship. Adding the Anti-Piracy Curtain to the P-Trap and Fire Hose countermeasures improves the success rate to 99.7%, but increased the system cost to a five-year cost of \$1.576M/ship.

THIS PAGE INTENTIONALLY LEFT BLANK

V. RECOMMENDATIONS

While multiple system configurations are effective for commercial vessels seeking to prevent pirate boarding, the research indicates that the P-Trap, supplemented with either Compressed Air Cannons or Fire Hoses is the best option for preventing maritime piracy based on the effectiveness, associated costs, ease of use, and the other determining factors. Because the Fire Hose countermeasure is a less complicated system that requires less crew action during a pirate boarding attempt, it is recommended that a combination of P-Traps and Fire Hoses be employed on commercial shipping vessels traversing Indonesian waters.

There are additional factors that were not modeled in this effort, such as levels of crew training, or employing countermeasures in a scenario where lookouts have failed to notice a pirate attack at range. These are areas for possible future work on boarding-prevention countermeasures. It should be noted that the P-Trap is a system that can be employed well in advance of a pirate attack. This allows the system to protect a vessel even in situations where crew lookouts do not notice pirate vessels approaching and system success is not dependent on crew training for successful use during a pirate attack.

THIS PAGE INTENTIONALLY LEFT BLANK

APPENDIX A. SYSTEM CONFIGURATIONS LIST

	Passive Defense		Active Defense		Active offense
Run #	P-Trap	Water Cannon	Razor Wire	Pirate Curtain	Compressed Air Cannon
1	off	off	On	off	off
2	On	off	On	off	On
3	On	On	On	off	On
4	On	On	On	On	off
5	On	off	On	On	off
6	off	off	On	On	On
7	On	On	On	off	off
8	On	off	off	On	off
9	off	off	On	On	off
10	On	On	On	On	On
11	On	On	off	off	off
12	On	off	On	On	On
13	On	On	off	On	off
14	On	off	off	off	off
15	off	On	On	On	On
16	On	off	On	off	off
17	On	On	off	On	On
18	On	On	off	off	On
19	off	off	On	off	On
20	off	On	On	off	off
21	On	off	off	off	On
22	On	off	off	On	On
23	off	On	On	On	off
24	off	On	On	off	On

THIS PAGE INTENTIONALLY LEFT BLANK

APPENDIX B. MODEL SOURCE CODE

A. AIR_CANNON.M

```
classdef Air_Cannon
    %Air_Cannon Represents the Air Cannon countermeasure countermeasure.
    %          the air cannon is a mounted, remote controlled turret that fires one of
several projectiles to stop pirate skiffs. Selected for the model from among these was
the net/line option that is fired at pirate craft to ensnare their propellers.

    properties
        max_range = 850; %Described the maximum range that the air cannon can fire.
Defined by
http://www.bcbin.com/products/product_details.php?category=marine&product=Security
Buccaneer#BUC001
        max_magazine_size = 6;  %the maximum number of shots that can be fired without
the countermeasure being reloaded.
        time_between_shots = 10;  %Delay, in seconds, that the cannon must wait between
successive shots.
        prob_of_hit = 0.5;  %Likelihood of any given shot successfully disabling a pirate
skiff.
        time_of_last_shot;  %Tracks the time at which the last shot was fired in order to
determine when the next shot is eligible to be fired.
        magazine;  %Tracks how many shots are remaining in the magazine for the current
run.
    end

    methods
        function obj = Air_Cannon()
            %Constructor allocates space and pointers for the object as well as
specifically initializing specific variables.
            obj.magazine = 0;
            obj.time_of_last_shot = 0;
        end

        function obj = activate(obj)
            %Turns the Air Cannon on by setting its magazine to full.
            obj.magazine = obj.max_magazine_size;
        end

        function [obj,skiffs] = act(obj,origin,skiffs,time)
            %Takes appropriate actions such as movement for the skiff for the given time
step as well as calling action for all its Pirates.

            %If there is ammunition remaining.
            if(obj.magazine)

                %If enough time has passed to allow another shot
                if(time - obj.time_of_last_shot  > obj.time_between_shots)

                    %Build an array of all Skiffs within range of the Air
                    %Cannon.
                    potential_targets = 0;
                    for i=1:size(skiffs,2)
                        if norm(skiffs(i).location-origin) < obj.max_range
                            if potential_targets
                                potential_targets(size(potential_targets,2)+1) = i;
                            else
                                potential_targets(1) = i;
                            end
                        end
                    end

                    %If any Skiffs were found within range.
                    if(potential_targets)

                        %select a target skiff from the potential targets
                        target = potential_targets(randi(size(potential_targets,2)));

                        %take the shot
                        shot = unifrnd(0,1);
                        if(shot < obj.prob_of_hit)
                            skiffs(target).status = Skiff_Status.DISABLED;
                        end

                        %reset timer
                        obj.time_of_last_shot = time;
                    end
                end
            end
        end

    end
end
```

B. BARBED_WIRE.M

```
classdef Barbed_Wire
    %Barbed_Wire Represents the Barbed Wire countermeasure countermeasure. Delays Pirates
in boarding.
    %          the countermeasure option of wrapping the perimeter of the ship with
barbed or razor wire is represented by the Barbed Wire class. The Barbed Wire object in
turn creates a large quantity of Barbed Wire Segment objects which each track the health
of the barbed wire over a small portion of the Commercial Ship’s circumference. Directly
adds to the time needed for Pirates to board.

    properties
        time_to_remove = 120; %Assumed time, in seconds, it would take a single pirate to
bypass or remove the barbed/razor wire barricade. Update with time in seconds it is
believed will be needed to bypass a length of razor wire.
        segment_size = 5; %number of degrees of the ship’s circle that constitute a
discrete section of razor wire. Psuedo-arbitrary at this time.
        segments = [Barbed_Wire_Segment(0,179,0),Barbed_Wire_Segment(-1,-180,0)];  %An
array of the Barbed Wire Segment sections to track each independently.
    end

    methods

        function obj = Barbed_Wire()
            %Constructor allocates space and addresses as well as initializing the Barbed
Wire segments to a disabled state.

            %Initialize Segments to broad sections with no durability.
            obj.segments = [Barbed_Wire_Segment(0,179,0),Barbed_Wire_Segment(-1,-180,0)];
        end

        function obj = Activate(obj)
            %Reinitializes the Barbed Wire Segment to segment sizes as defined by the
segment_size property and with appropriate durability.

            %Iterate through all segments on on half of the ship
            for i=1:(180/obj.segment_size)
                obj.segments(i) = Barbed_Wire_Segment(obj.segment_size*(i-
1),(obj.segment_size)*i - 1, obj.time_to_remove);
            end
            %Iterate through the other side of the ship
            for j = ((180/obj.segment_size)+1):((180/obj.segment_size)*2)
                obj.segments(j) = Barbed_Wire_Segment((obj.segment_size*(j-i-1)+1)*(-
1),(obj.segment_size)*(j-i)*(-1) , obj.time_to_remove);%(obj.segment_size)*(i-1)*(-
1),((obj.segment_size)*i - 1)*(-1), obj.time_to_remove);
            end
        end

        function durability = is_effective(obj,relative_approach_vector)
            %Tests whether there is barbed wire countermeasure remaining in a particular
direction.
```

```
            relative_angle = get_angle([1,0],relative_approach_vector);
            durability = 0;

            %Find the appropriate Barbed Wire Segment and test its
            %durability.
            for i = 1:size(obj.segments,2)
                if(relative_angle < obj.segments(i).max_angle) && (relative_angle > 
obj.segments(i).min_angle)
                    durability = obj.segments(i).durability;
                    break;
                end
            end
        end

        function obj = degrade(obj, time_spent_bypassing, relative_approach_vector)
            %Reduces the durability in the Barbed Wire Segment at the appropriate area.

            relative_angle = get_angle([1,0],relative_approach_vector);

            %Find appropriate segment and reduce its durability.
            for i = 1:size(obj.segments,2)
                if(relative_angle < obj.segments(i).max_angle) && (relative_angle > 
obj.segments(i).min_angle)
                    obj.segments(i) = obj.segments(i).degrade(time_spent_bypassing);
                    break;
                end
            end
        end

    end

end
```

C. BARBED_WIRE_SEGMENT.M

```
classdef Barbed_Wire_Segment
    %Barbed_Wire_Segment Class that defines a small length of the Barbed Wire
countermeasure. Used to track health for an arbitrarily small portion of the overall
barricade.
    %   the Barbed Wire Segment is a small part of what makes up the entire Barbed Wire
barricade. Since the barrier can be broken at any particular point, the condition must be
tracked in segments to show how a breach can happen at one point due to focused efforts.

    properties
        min_angle; %Defines the starting point along the unit circle of this segment.
        max_angle; %Defines the ending point along the unit circle of this segment.
        durability; %Defines how many seconds it would take an individual to bypass or
remove the segment. Serves as the overall health of the segment.
    end

    methods
        function obj = Barbed_Wire_Segment(min, max, time_to_bypass)
            %Constructor that allocates memory and addresses as well as initializes
object variables to the selected parameters.

            obj.min_angle = min;
            obj.max_angle = max;
            obj.durability = time_to_bypass;
        end

        function obj = degrade(obj, time_spent_bypassing)
            %Takes in the time a Pirate has spent working against the
            %Segment and updates the Segment’s durability.

            obj.durability = obj.durability - time_spent_bypassing;
        end

        function remaining_durability = is_functional(obj)
            %Query to determine if the particular Segment can still hold back a Pirate.

            remaining_durability = 0;
            if(obj.durability > 0)
                remaining_durability = obj.durability;
            end
        end

    end

end
```

D. COM_SHIP.M

```
classdef Com_Ship
    %Com_Ship Class representing the Commercial Ship that is targeted by Pirates within
the scenario.
    %          the Commercial Ship class represents the target ship from the DRM within
the model. It tracks the location, velocity and all other necessary factors associated
with the ship's status and actions. The Commercial Ship object contains within itself
objects representing all of its crew members as well as the objects associated with each
countermeasure.

    properties
        move_speed = 10.28; %the maximum movement speed of the Commercial Ship. Based on
presumed 20 knot max speed. Needs to be verified. representative value based on the
design scenario
        location = [0,0];  %Holds the current location of the Commercial Ship within the
model space.
        crew;  %A 1 x n array holding Crew objects to represent the Commercial Ship's
crew members.
        bearing;  %A unit vector showing which direction the ship traveled in the
previous time increment in order to know the direction it is oriented.
        last_time;  %Tracks when the object was last updated within model time in order
to scale action progress appropriately.
        ptrap_on; %Flag indication if the P-Trap is engaged in the current run.
        ptrap; %Object representing the P-Trap countermeasure within the model.
        water_cannons_on;  %Flag ingicationg if the Water Cannons are present and engaged
in the current model run.
        water_cannons = Water_Cannon(0,0); %1 x n array of objects representing the Water
Cannons within the model.
        wire_perimeter = Barbed_Wire(); %Object representing the ship's Barbed Wire
countermeasure.
        pirate_curtain = Curtain(); %Object representing the Pirate Curtain
countermeasure within the model.
        air_cannon = Air_Cannon(); %Object representing the Air Cannon countermeasure
within the model.
    end

    methods
        function obj = Com_Ship(initial_position, crew_status, ptrap_enabled,
water_cannon_enabled, wire_perimeter_enabled, pirate_curtain_enabled,air_cannon_enabled)
            %Constructor allocates memory for the object and establishes its pointers as
well as initializing object properties and constructing the countermeasure objects.

            %Initialize properties
            obj.location = initial_position;
            obj.crew =
[Crew(crew_status(1)),Crew(crew_status(2)),Crew(crew_status(3)),Crew(crew_status(4)),Crew
(crew_status(5)),Crew(crew_status(6)),Crew(crew_status(7)),Crew(crew_status(8))]; %8 is
an arbitrary max. Fill out DRM to define crew sizes
            obj.last_time = 0;

            %Construct and initialize countermeasures as defined by the
            %input parameter flags.
            if ptrap_enabled
                obj.ptrap_on = ptrap_enabled;
                obj.ptrap = Ptrap();
            end

            if water_cannon_enabled
                obj.water_cannons(1) = Water_Cannon(0,60);
                obj.water_cannons(2) = Water_Cannon(60,120);
                obj.water_cannons(3) = Water_Cannon(120,180);
                obj.water_cannons(4) = Water_Cannon(-60,0);
                obj.water_cannons(5) = Water_Cannon(-120,-60);
                obj.water_cannons(6) = Water_Cannon(-180,-120);
                obj.water_cannons_on = water_cannon_enabled;
            end

            if wire_perimeter_enabled
                obj.wire_perimeter = obj.wire_perimeter.Activate();
            end

            if pirate_curtain_enabled
                obj.pirate_curtain = obj.pirate_curtain.activate();
            end

            if air_cannon_enabled
                obj.air_cannon = obj.air_cannon.activate();
            end
        end

        function loc  = getLocation(obj)
            %Returns the current location of the Commercial Ship within the model space.

            loc = obj.location;
        end

        function crew_state = getCrewStatus(obj)
            %Returns the a copy of Crew array

            crew_state = obj.crew;
        end

        function obj = move(obj, time, military_ship_loc)
            %Moves the commercial ship closer to the military ship based on the move
speed is used to scale the magnitude.

            % Calculate the directional vector
            vector = [military_ship_loc(1)-obj.location(1), military_ship_loc(2)-
obj.location(2)];

            % Normalize the vector and scale it to both the ship's speed
            % and the time increment
            vector = vector/norm(vector);
            obj.bearing = vector;
            vector = vector * obj.move_speed * (time - obj.last_time);

            % Update x,y and time records
            obj.location(1) = obj.location(1) + vector(1);
            obj.location(2) = obj.location(2) + vector(2);
            obj.last_time = time;
        end

        function [obj,skiffs] = act(obj, time, skiffs)
            %Causes the Commercial Ship to act in response to the actions of the Skiffs.
Includes effects of the Ship countermeasures on the SKiffs.

            %Cause all able Crew members to act.
            for i = 1:size(obj.crew,2)
                if((obj.crew(i).status ~= Crew_Status.ABSENT)&&(obj.crew(i).status ~=
Crew_Status.KILLED)) %This elimination statement may be replaced by switch case within
Crew
                    obj.crew(i).act(time,skiffs);
                end
            end

            %Cause all active countermeasures to act.
            if obj.ptrap_on
               [obj.ptrap,skiffs] = obj.ptrap.ptrap_effect(obj,skiffs);
            end

            if obj.water_cannons_on
                for n = 1:6
                    [obj.water_cannons(n),skiffs] =
obj.water_cannons(n).Engage(obj,skiffs);
                end
            end

            [obj.air_cannon,skiffs] = obj.air_cannon.act(obj.location,skiffs,time);
        end
    end

end
```

E. CREW.M

```
classdef Crew
    %Crew Class represents an individual crew member of the targeted commercial ship.
    %   the Crew class objects represent individual crew members of the targeted
commercial ship. Initially they were intended for more complex behavior and that may be
spiraled in during future efforts, but for this model increment the class serves largely
as a placeholder.

    properties
        status; %Enumerated indicator of the Crew member's current status.
    end

    methods
        function obj = Crew(initial_status)
            %Constructor allocates memory and address pointers as well as initializing
the status.

            obj.status = initial_status;
        end

        function obj = act(obj,time,skiffs)
            %Logic defining how individual crew members would act in each time
incremement can be inserted here.
        end
    end

end
```

F. CREW_STATUS.M

```
classdef Crew_Status
    %Crew_Status Enumeration of possible states for a Crew object.
    %   Enumeration class describing the possible states for a Crew member to be in.

    enumeration
        ABSENT
        KILLED
        IDLE
    end

end
```

G. CURTAIN.M

```
classdef Curtain
    %Curtain Represents the Pirate Curtain countermeasure countermeasure.
    %          the commercially advertised pirate curtain system consists of a
combination of fire hoses used to flood pirate skiffs and erratically flailing hoses with
weighted ends that can cause bodily harm to individuals scaling the side of the vessel.
It was determined that the first component of the system heavily overlapped with the
Water Cannon already under consideration, but that the flail version represented a unique
countermeasure option. The Pirate Curtain class then represents the flail portion alone
of the commercially proposed solution.       the Object monitors the Port and Starboard
regions of the Commercial Ship and applies a chance to strike any pirate who is in the
process of attempting to board. If struck, the Pirate is presumed to be permanently
disabled within the timeline of the scenario.

    properties
        per_second_probability_of_impacting_pirate = 0.05; %Defines the likelihood in
each second interval that any pirate within range of the flailing hose would be struck
solidly. Studies are needed to determine better effectiveness data
        min_angle = 20; %the start of the angular range over which the Pirate Curtain
covers one side of the ship.
        max_angle = 160; %the ending angle of the angular range over which the Pirate
Curtain covers on side of the ship.
        curtain_active = 0; %Flag indicatiing if the Pirate Curtain is present and
active.
    end

    methods
        function obj = Curtain()
            %Constructor allocates memory and establishes addess pointers. Sets validity
flag to FALSE until the countermeasure is initialized.

            obj.curtain_active = 0;
        end

        function obj = activate(obj)
            %Initializes the Pirate Curtain by setting the validity flag to TRUE.

            obj.curtain_active = 1;
        end

        function pirate = flail(obj,pirate,approach_vector,time)
            %Simulates the Pirate Curtain's flailing action over the time leading up to
the input time point in order to determine if any Pirates were struck and disable them if
so.

            approach_angle = get_angle([1,0],approach_vector);

            %Determine if the Pirate is within the Pirate Curtain's reach
            if(((approach_angle > obj.min_angle)&&(approach_angle <
obj.max_angle))||((approach_angle < (-1)*obj.min_angle)&&(approach_angle>(-
1)*obj.max_angle)))
```

```
                cumulative_probability_of_survival = power((1-
obj.per_second_probability_of_impacting_pirate),time - pirate.time_entering_curtain);

                %randomly determine if pirate is struck
                if(unifrnd(0,1) < cumulative_probability_of_survival)
                    %pirate survives
                    pirate.time_entering_curtain = time;
                else
                    %pirate is struck and disabled
                    pirate.status = Pirate_Status.KILLED;
                end
            end
        end
    end

end
```

H. DISPLAY.M

```
classdef Display
    %Display Controls all display functionaility for the model system.
    %   Class object used to implement a model display feature for testing and
demonstrations of the model.

    properties
        width; %Defines the window size for the display view.
    end

    methods
        function obj = Display(skiff_start_range)
            %Constructor establishes memory space and addresses for the Display object.

            obj.width = 2 * skiff_start_range;
        end

        function obj = refresh(obj,com_ship, mil_ship, skiffs)
            %Resets all element positions within the view screen

            % set the vertex associated with the commercial ship
            scatter_x = com_ship.location(1);
            scatter_y = com_ship.location(2);
            scatter_size = 20;
            scatter_color = [0,0,1];

            % add the vertex associated with the military ship
            scatter_x(2) = mil_ship.location(1);
            scatter_y(2) = mil_ship.location(2);
            scatter_size(2) = 20;
            scatter_color(2,:) = [0,1,0];
            % add vertices for all skiffs
            for i=1:size(skiffs,2)
                scatter_x(2+i) = skiffs(i).location(1);
                scatter_y(2+i) = skiffs(i).location(2);
                scatter_size(2+i) = 10;
                switch skiffs(i).status
                    case Skiff_Status.NORMAL
                        scatter_color(2+i,:) = [1,0,0];
                    case Skiff_Status.DISABLED
                        scatter_color(2+i,:) = [0.3,0.3,0.3];
                    case Skiff_Status.REPELLED
                        scatter_color(2+i,:) = [0.5,0,0];
                    otherwise
                        scatter_color(2+i,:) = [0,0,0];
                end
            end


            %display plot
            scatter(scatter_x,scatter_y,scatter_size,scatter_color);

            %scale plot
            axis([com_ship.location(1) - obj.width/2, com_ship.location(1) + obj.width/2,
com_ship.location(2) - obj.width/2, com_ship.location(2) + obj.width/2]);

        end
    end

end
```

I. DOMAIN_MANAGER.M

```
classdef Domain_Manager
    %Domain_Manager controls overall operation of the model.
    %           the Domain Manager object controls operation of the overall model. It
creates the other object within itself, initializes them, and then calls their operations
iteratively to simulate the passage of time. Domain Manager controls the passing of data
between the other objects (as opposed to a shared memory structure). Domain Manager
determines when a scenario has concluded based upon established criteria; primarily that
all pirates have been disabled or that one has boarded. The Domain Manager also controls
repeated runs of the various system configurations in order to calculate the overall MOE.
Domain Manager is the connection between the MATLAB user interface and the rest of the
program, it takes in the scenario configuration and outputs the calculate probability of
survival for the commercial ship.

    properties
        time_increment = 1;  %Controls how many seconds pass between each time increment
of the model. Decrease to improve fidelity and increase to improve run speed.
        distance_to_aid = 1075000; %Sets the distance between the targeted Commercial
Ship and the Military Ship that can aid it. Jakarta and Belawan appear to be the two most
distant naval bases at 2154 km from each other. Halfway between them would be 1075km.
        skiff_start_range = 8046; %Sets how far away the skiffs are when the scenario
starts. Based on binocular range described in
https://answers.yahoo.com/question/index?qid=20120516151214AAieTBM
        min_num_skiffs = 6;  %Minimum number of skiffs that will appear in any run of the
model.
        max_num_skiffs = 10;  %Maximum number of skiffs that will appear in any run of
the model.
        num_skiffs = 0;  %Holds the number of skiffs in the current run of the model.
        mil_ship  %Mil_Ship class object that represents the Military Ship coming to the
Commercial Ship’s aid in the scenario.
        com_ship  %Com_Ship object that represents the targeted Commercial Ship the
pirates are attempting to overwhelm.
        skiffs  %A 1 x n array of Skiff objects that represents the swarm of pirate craft
attacking the COmmercial Ship.
        time  %Holds the current time, in seconds, since the beggining of the scenario.
    end

    methods
        function obj = Domain_Manager()
            %Simple constructor for Domain Manager. Allocates space and pointers for all
class members.
        end


        function results = SingleRun(obj,distance_to_aid, ptrap_enabled,
water_cannon_enabled,wire_perimeter_enabled,pirate_curtain_enabled,air_cannon_enabled)
            %Executes a single run of the selected scenario.

            %initialize time
            obj.time = 0;
            %initialize Commercial ship
            obj.com_ship = Com_Ship([1,1],[1,1,1,1,1,1,1,1],ptrap_enabled,
water_cannon_enabled, wire_perimeter_enabled, pirate_curtain_enabled,air_cannon_enabled);

            %initialize Military ship
            obj.mil_ship = Mil_Ship([1+distance_to_aid,1]);

            %initialize pirate skiffs
            obj.num_skiffs = randi([obj.min_num_skiffs, obj.max_num_skiffs],1);

            obj.skiffs = Skiff([0,0],[0,0,0,0,0,0,0]);
            for n = 1:obj.num_skiffs
                %place each skiff in a random direction from the commercial
                %ship
                direction = randn(1,2);
                direction = direction/norm(direction);
                displacement = direction * obj.skiff_start_range;
                obj.skiffs(n) =
Skiff([obj.com_ship.location(1)+displacement(1),obj.com_ship.location(2)+displacement(2)]
,[1,1,1,1,1,1,1]); %need random position at range and table input if arms
            end

            %keep track of how many skiffs are still active so the scenario
            %can be aborted if they are all disabled
            active_skiff_count = size(obj.skiffs,2);

            %iterate scenario
            while ((sqrt((obj.com_ship.location(1)-obj.mil_ship.location(1))^2 +
(obj.com_ship.location(2)-obj.mil_ship.location(2))^2) > ((obj.com_ship.move_speed +
obj.mil_ship.move_speed)*obj.time_increment))&&(~CountBoarders(obj.skiffs))&&active_skiff
_count)
                %update time
                obj.time = obj.time + obj.time_increment;

                %move the commercial ship
                obj.com_ship = obj.com_ship.move(obj.time, obj.mil_ship.getLocation());

                %move the military ship
                obj.mil_ship = obj.mil_ship.move(obj.time, obj.com_ship.getLocation());

                %All skiffs act
                for i = 1:obj.num_skiffs
                    [obj.skiffs(i),obj.com_ship] =
obj.skiffs(i).act(obj.time,obj.com_ship);
                end

                %Commercial ship's crew acts
                [obj.com_ship,obj.skiffs] = obj.com_ship.act(obj.time,obj.skiffs);


                %update number of active skiffs
                active_skiff_count = 0;
                for h=1:size(obj.skiffs,2)
```

```
                    if obj.skiffs(h).status ~= Skiff_Status.DISABLED
                        active_skiff_count = active_skiff_count + 1;
                    end
                end

            end


            %Return TRUE if no boarders made it onto the ship, FALSE if any
            %boarders are on the ship.
            if (~CountBoarders(obj.skiffs))
                results = 1;
            else
                results = 0;
            end
        end


        function results = DemoRun(obj, ptrap_enabled,
water_cannon_enabled,wire_perimeter_enabled,pirate_curtain_enabled,air_cannon_enabled)
            %Executes a single run of the model with the specified
            %parameters and displaying a graphic of the ships and skiffs
            %moving.

            %the Display object manages the view window to show the
            %commercial ship, military ship and skiffs as moving dots on a
            %chart.
            viewer = Display(obj.skiff_start_range);

            %initialize time
            obj.time = 0;

            %initialize Commercial ship
            obj.com_ship = Com_Ship([1,1],[1,1,1,1,1,1,1,1],ptrap_enabled,
water_cannon_enabled, wire_perimeter_enabled, pirate_curtain_enabled,air_cannon_enabled);

            %initialize Military ship
            obj.mil_ship = Mil_Ship([1+obj.distance_to_aid,1]);

            %initialize pirate skiffs
            obj.num_skiffs = randi([obj.min_num_skiffs, obj.max_num_skiffs],1);

            obj.skiffs = Skiff([0,0],[0,0,0,0,0,0,0]);
            for n = 1:obj.num_skiffs
                %place each skiff in a random direction from the commercial
                %ship
                direction = randn(1,2);
                direction = direction/norm(direction);
                displacement = direction * obj.skiff_start_range;
                obj.skiffs(n) =
Skiff([obj.com_ship.location(1)+displacement(1),obj.com_ship.location(2)+displacement(2)]
,[1,1,1,1,1,1,1]); %need random position at range and table input if arms
            end
            %iterate scenario
            while ((sqrt((obj.com_ship.location(1)-obj.mil_ship.location(1))^2 +
(obj.com_ship.location(2)-obj.mil_ship.location(2))^2) > ((obj.com_ship.move_speed +
obj.mil_ship.move_speed)*obj.time_increment))&&(~CountBoarders(obj.skiffs)))
                %update time
                obj.time = obj.time + obj.time_increment;

                %move the commercial ship
                obj.com_ship = obj.com_ship.move(obj.time, obj.mil_ship.getLocation());

                %move the military ship
                obj.mil_ship = obj.mil_ship.move(obj.time, obj.com_ship.getLocation());

                %All skiffs act
                for i = 1:obj.num_skiffs
                    [obj.skiffs(i),obj.com_ship] =
obj.skiffs(i).act(obj.time,obj.com_ship);
                end

                %Commercial ship's crew acts
                [obj.com_ship,obj.skiffs] = obj.com_ship.act(obj.time,obj.skiffs);

                %update plot in each increment
                viewer = viewer.refresh(obj.com_ship,obj.mil_ship,obj.skiffs);
                drawnow;
            end

            %Return TRUE if no boarders made it onto the ship, FALSE if any
            %boarders are on the ship.
            if (~CountBoarders(obj.skiffs))
                results = 1;
            else
                results = 0;
            end
        end


        function results = Multiple_Trials(obj,num_of_runs, ptrap_enabled,
water_cannon_enabled,wire_perimeter_enabled,pirate_curtain_enabled,air_cannon_enabled)
            %Runs the model with the specified parameters the specified
            %number of times and then returns the proportion of times the
            %skiff avoided being boarded.

            %Establish a counter to track the number of times the
            %commercial ship survives.
            num_of_survivals = 0;

            %Iterate single runs the specified number of times.
            for i=1:num_of_runs
                num_of_survivals = num_of_survivals + obj.SingleRun(obj.distance_to_aid,
ptrap_enabled,
water_cannon_enabled,wire_perimeter_enabled,pirate_curtain_enabled,air_cannon_enabled);
```

```
            end

            %Calculate the proportion of survivors.
            results = num_of_survivals/num_of_runs;
        end

    end

end

function has_boarders = CountBoarders(skiffs)
%Helper function that takes in an array of Skiffs and counts the number of
%Pirates marked as having boarded the ship.
has_boarders = 0;
for i = 1:size(skiffs,2)
    for j = 1:size(skiffs(i).pirates,2)
        if(skiffs(i).pirates(j).status == Pirate_Status.BOARDED)
            has_boarders = has_boarders + 1;
        end
    end
end
end
```

J. GET_ANGLE.M

```
function [ angle_in_degrees ] = get_angle( a,b )
%get_angle Helper function determines the angle in between two directional vectors.
%   Helper function determines the angle in between two directional vectors.

angle_in_degrees = acosd(dot(a,b)/(norm(a)*norm(b)));

anticlockwise = cross([a,0],[b,0]);
anticlockwise = anticlockwise(3);

if anticlockwise < 0
    angle_in_degrees = angle_in_degrees * (-1);
end
end
```

K. MIL_SHIP.M

```
classdef Mil_Ship
    %Mil_Ship Represents the Military Ship that is coming to the target commercial
vessel's aid within the model.
    %          the Military Ship class generically represents some assisting vessel
coming to the aid of the Commercial Ship that can stop the pirate attack if it arrives in
time. The program object actually does very little, merely tracking its own progression.
If it reaches the Commercial Ship then Domain Manager will end the scenario.

    properties
        move_speed = 15.55; %the maximum speed at which the Military SHip may move.
Representative value based on the design scenario; based on advertised speed of Arleigh-
Burke class destroyer
        location = [0,0];  %the current locatgion of the Military SHip within the model
space.
        bearing;  %the direction in which the ship is facing as a vector.
        last_time;  %the last point, in model time, at which the Military Ship object was
updated.
    end

    methods
        function obj = Mil_Ship(initial_position)
            %Constructor allocates memory, sets address pointers and initializes object
properties to the starting values.

            obj.location = initial_position;
            obj.last_time = 0;
        end

        function loc  = getLocation(obj)
            %Returns the current location of the Military Ship within the %model space.

            loc = obj.location;
        end

        function obj = move(obj, time, commercial_ship_loc)
            %Moves the military ship closer to the commercial ship based on the move
speed is used to scale the magnitude.

            % Calculate the directional vector
            vector = [commercial_ship_loc(1)-obj.location(1), commercial_ship_loc(2)-
obj.location(2)];

            % Normalize the vector and scale it to both the ship's speed
            % and the time increment
            vector = vector/norm(vector);
            obj.bearing = vector;
            vector = vector * obj.move_speed * (time - obj.last_time);

            % Update x,y and time records
            obj.location(1) = obj.location(1) + vector(1);
            obj.location(2) = obj.location(2) + vector(2);
            obj.last_time = time;
        end
    end

end
```

L. PIRATE.M

```
classdef Pirate
    %Pirate Represents the individual Pirate within the model.
    %          the Pirate object represents an individual pirate. A Pirate can transition
primarily between different states and keeps track of the time it should be taking to
accomplish tasks, such as boarding the side of a Commercial Ship.  Pirate objects are
created and reside within Skiff objects. The Skiff passes update calls down to the Pirate
object as well as all data or pointers needed for it to accomplish its tasks.



    properties
        weapon; %Enumeration representing the weapon this Pirate carries. Not used within
this version of the model.
        status; %Enumeration representing the current state of this individual Pirate.
        last_time; %the last time, in model time, that the Pirate object was updated.
Used to scale actions.
        task_start; %the time, in model time, that the Pirate’s current task was begun.
Used to calculate progress to completion.
        maximum_boarding_distance = 2; %the distance, in meters, that the Pirate must get
to the Commercial Ship in order to attempt to board it.
        max_simultaneous_boarders = 2; %the number of Pirates in one skiff who may
attempt to board the Commercial Ship at the same time. Driven by assumptions about the
stability of the skiff and the length of its rail.
        time_to_board = 45;            %the time, in seconds, it is expected to take a
pirate to board the Commercial Ship in the absence of any countermeasures. Considered a
strong Assumption
        time_entering_curtain = 0;  %the model time at which the Pirate entered the
Pirate Curtain’s region of influence.
    end

    methods
        function obj = Pirate(weapon_setting)
            %Constructor allocates memory, sets address pointers and sets the status of
the Pirate.

            obj.weapon = weapon_setting;
            obj.status = Pirate_Status.ABSENT;

            %If pirate was not entered as absent change the status to idle
            if obj.weapon ~= Pirate_Status.ABSENT
```

```
                obj.status = Pirate_Status.IDLE;
            end
            obj.last_time = 0;
        end


        function [obj,com_ship] = act(obj, time, skiff, com_ship, approach_vector)
            %Causes the Pirate to take action based on the Skiffs, Commercial Ship and
time passed since its last action.

            %Route to the appropriate logic based on the Pirate’s current
            %status
            switch obj.status
                case Pirate_Status.ABSENT
                    %no effect for absent pirate
                case Pirate_Status.KILLED
                    %no effect for dead pirate
                case Pirate_Status.IDLE
                    %Pirater assumes an action

                    %if the skiff is near the commercial ship the pirate
                    %may attempt to board
                    if(pdist([skiff.location;com_ship.getLocation()]) <
obj.maximum_boarding_distance)
                        %presuming that the skiff can only manage to let
                        %some of it’s passengers mount hooks/ladders at the
                        %same time it is necessary to count how many are
                        %already trying and wait till they are done
                        boarders = 0;
                        for i = 1:size(skiff.pirates,2)
                            if (skiff.pirates(i).status ==
Pirate_Status.ATTEMPTING_TO_BOARD)
                                boarders = boarders + 1;
                            end
                        end

                        %if there are not too many pirates attempting to
                        %board already, this pirate begins to attempt to
                        %board
                        if (boarders < obj.max_simultaneous_boarders)
                            obj.status = Pirate_Status.ATTEMPTING_TO_BOARD;
                            obj.task_start = time;
                            obj.time_entering_curtain = time;
                        end
                    end

                case Pirate_Status.PILOTING_SKIFF
                    %Pirate who is piloting cannot take other actions\
                case Pirate_Status.ATTEMPTING_TO_BOARD
                    %Pirates who are in the process of boarding

                    %If the barbed wire is in place the Pirate will spend
                    %time disabling it.
                    if(com_ship.wire_perimeter.is_effective(approach_vector))
                        com_ship.wire_perimeter = com_ship.wire_perimeter.degrade(time -
obj.task_start, approach_vector);
                        obj.task_start = time;
                    end

                    %If the Pirate Curtain is active then check to see if
                    %the Pirate is struck.
                    if com_ship.pirate_curtain.curtain_active
                        obj = com_ship.pirate_curtain.flail(obj,approach_vector,time);
                    end

                    %Check if the pirate has been working at boardinf
                    %sufficiently long.
                    if(time - obj.task_start > obj.time_to_board)
                        obj.status = Pirate_Status.BOARDED;
                    end
                case Pirate_Status.BOARDED
                    %If the Pirate has boarded the Domain Manager will end
                    %the scenario.
                otherwise
                    disp('invalid pirate status');
            end
            obj.last_time = time;
        end
    end

end
```

M. PIRATE_STATUS.M

```
classdef Pirate_Status
    %Pirate_Status Enumeration describing the possible states a Pirate may be in.
    %   Enumeration describing the possible states a Pirate may be in.

    enumeration
        ABSENT
        KILLED
        IDLE
        SMALL_ARM
        RPG
        PILOTING_SKIFF
        ATTEMPTING_TO_BOARD
        BOARDED
    end

end
```

N. PTRAP.M

```
classdef Ptrap
    %Ptrap Represents the Pirate Trap countermeasure countermeasure within the model.
    %          the pirate trap countermeasure is a system of difficult to see lines
trailed through the water along the sides of and behind the commercial ship in order to
foul the propellers of pirate craft. Three seperate regions (port, starboard, aft) are
seperately tracked as they each have their own set of lines.

    properties
        port_line_count; %Number of lines currently in the port region.
        starboard_line_count; %Number of lines currently within the starboard region.
        aft_line_count; %Number of lines currently in the aft region.
        num_of_lines = 10; %Number of lines all sectors will be initialized with.
        ptrap_range = 10; %the distance (m) from the Commercial Ship hull that the P-Trap
lines extend out.
        port_min_angle = 30; %the starting angle for the port region.
        port_max_angle = 150; %Ending angle for the port region.
        starboard_min_angle = -150; %Starting angle for the starboard region.
        starboard_max_angle = -30; %Ending angle for the starboard region.
        aft_min_angle = -150; %Starting angle for the aft region.
        aft_max_angle = 150; %Ending angle for the aft region.

    end

    methods

        function obj = Ptrap()
            %Constructor allocates memory and sets the starting number of lines into all
three regions.

            obj.port_line_count = obj.num_of_lines;
            obj.starboard_line_count = obj.num_of_lines;
            obj.aft_line_count = obj.num_of_lines;
        end

        function [obj,skiffs] = ptrap_effect(obj,com_ship,skiffs)
            %Causes any appropriate effects on the Skiffs to be effected.

            %Check all skiffs to see if they lie within one of the active
            %regions.
            for i = 1:size(skiffs,2)
                %starboard
                if (obj.starboard_line_count > 0)&&
(pdist([com_ship.location;skiffs(i).location],'euclidean') < obj.ptrap_range) &&
(get_angle(com_ship.bearing,skiffs(i).location - com_ship.location) >
obj.starboard_min_angle) && (get_angle(com_ship.bearing,skiffs(i).location -
com_ship.location) < obj.starboard_max_angle)
                    skiffs(i).status = Skiff_Status.DISABLED;
                    obj.starboard_line_count = obj.starboard_line_count - 1;
                end
                %port
                if (obj.port_line_count > 0)&&
(pdist([com_ship.location;skiffs(i).location],'euclidean') < obj.ptrap_range) &&
(get_angle(com_ship.bearing,skiffs(i).location - com_ship.location) > obj.port_min_angle)
&& (get_angle(com_ship.bearing,skiffs(i).location - com_ship.location) <
obj.port_max_angle)
                   skiffs(i).status = Skiff_Status.DISABLED;
                   obj.port_line_count = obj.port_line_count - 1;
                end
                %aft
                if (obj.aft_line_count > 0)&&
(pdist([com_ship.location;skiffs(i).location],'euclidean') < obj.ptrap_range) &&
((get_angle(com_ship.bearing,skiffs(i).location - com_ship.location) > obj.aft_max_angle)
|| (get_angle(com_ship.bearing,skiffs(i).location - com_ship.location) <
obj.aft_min_angle))
                   skiffs(i).status = Skiff_Status.DISABLED;
                   obj.aft_line_count = obj.aft_line_count - 1;
                end
            end
        end
    end

end
```

O. SKIFF.M

```
classdef Skiff
    %Skiff Represents a single pirate skiff within the model.
    %          the Skiff class represents the pirate vessels within the model. The object
tracks the stats, position, and crew of a particular skiff and determines the skiff's
next actions when called. Each Skiff object also contains a number of Pirate objects
(described below) and triggers them to carry out their own actions in each time
increment. The Skiff objects are all generated and tracked by Domain Manager.     In
general, Skiffs move directly towards the Commercial Ship in an attempt to allow their
Pirates to board. Skiffs can be disabled by countermeasures, rendering them immobile. If
all Skiffs in the scenario are disabled the Domain Manager will end the scenario.

    properties
        location; %the current location in the model space of the pirate Skiff.
        pirates; %1 x n array of Pirates aboard this SKiff.
[pirat1,pirate1,pirate2,...,pirate7]
        move_speed = 12.86; %Maximum movement speed of the Skiff.  25 knots max speed (in
meters per second) based upon literature
        bearing; %Direction the Skiff is currently facing.
        status; %Current state of the Skiff.
        last_time; %the model time at which the SKiff was last updated.
        approach_vector; %Stores the relative direction the skiff is from the commercial
ship so that when they overlap in the model the skiff can still be treated as having come
to some particular side of the ship
    end

    methods
        function obj = Skiff(initial_position,pirate_settings)
            %Constructor allocates memory for members, initializes properties, and
generates all Pirate objects.

            obj.location = initial_position;
            obj.pirates = [Pirate(pirate_settings(1)), Pirate(pirate_settings(2)),
Pirate(pirate_settings(3)), Pirate(pirate_settings(4)), Pirate(pirate_settings(5)),
Pirate(pirate_settings(6)), Pirate(pirate_settings(7))];
            obj.last_time = 0;
            obj.status = Skiff_Status.NORMAL;
        end

        function [obj, com_ship] = act(obj, time, com_ship)
            %Updates the skiff to include approaching/moving with the ship as well as
activating any passenger actions.

            %determine if the skiff has a valid pilot
            has_pilot = false;
            for i=1:size(obj.pirates,2)
                if obj.pirates(i).status == Pirate_Status.PILOTING_SKIFF
                    has_pilot = true;
                end
            end
            % if it does then the skiff moves towards the ship
            if has_pilot
                obj = obj.move(time, com_ship.getLocation());
            else
                %if not look for an idle pirate to take over piloting
                for j=1:size(obj.pirates,2)
                    if obj.pirates(j).status == Pirate_Status.IDLE
                        obj.pirates(j).status = Pirate_Status.PILOTING_SKIFF;
                        has_pilot = true;
                        break;
                    end
                end
                %if no idle pirates, pull one from another task
                if(~has_pilot)
                    for k = 1:size(obj.pirates,2)
                        if (obj.pirates(k).status ~= Pirate_Status.ABSENT) &&
(obj.pirates(k).status ~= Pirate_Status.KILLED)
                            obj.pirates(k).status = Pirate_Status.PILOTING_SKIFF;
                            has_pilot = true;
                            break;
                        end
                    end
                end
            end

            % All Pirates take action
            for m = 1:size(obj.pirates,2)
                [obj.pirates(m),com_ship] = obj.pirates(m).act(time, obj, com_ship,
obj.approach_vector);
            end

            %update time
            obj.last_time = time;
        end

        function obj = move(obj, time, target_loc)
            %calculates the movement in each time increment. An increment is lost each
time the pilot must be changed in the current schema. May want to add
acceleration/deacceleration effects.

            % Calculate the directional vector
            vector = [target_loc(1)-obj.location(1), target_loc(2)-obj.location(2)];

            % Normalize the vector and scale it to both the ship's speed
            % and the time increment
            vector = vector/norm(vector);
            obj.bearing = vector;
            vector = vector * obj.move_speed * (time - obj.last_time);

            if obj.status == Skiff_Status.NORMAL

                %if distance to commercial ship is greater than the distance
                %the skiff can move, then move as far as it can; otherwise move
                %to the ship's point.
                if(pdist([obj.location;target_loc],'euclidean') > (obj.move_speed * (time
- obj.last_time)))
                    % Update x,y and time records
                    obj.location(1) = obj.location(1) + vector(1);
                    obj.location(2) = obj.location(2) + vector(2);
                    obj.approach_vector = -vector;
                else
                    obj.location(1) = target_loc(1);
                    obj.location(2) = target_loc(2);
                end

            elseif obj.status == Skiff_Status.REPELLED
                %If the Skiff is being repelled by a countermeasure it
                %heads in the opposite direction.
                    obj.location(1) = obj.location(1) - vector(1);
                    obj.location(2) = obj.location(2) - vector(2);
                    obj.approach_vector = -vector;
            end

        end

    end

end
```

P. SKIFF_STATUS.M

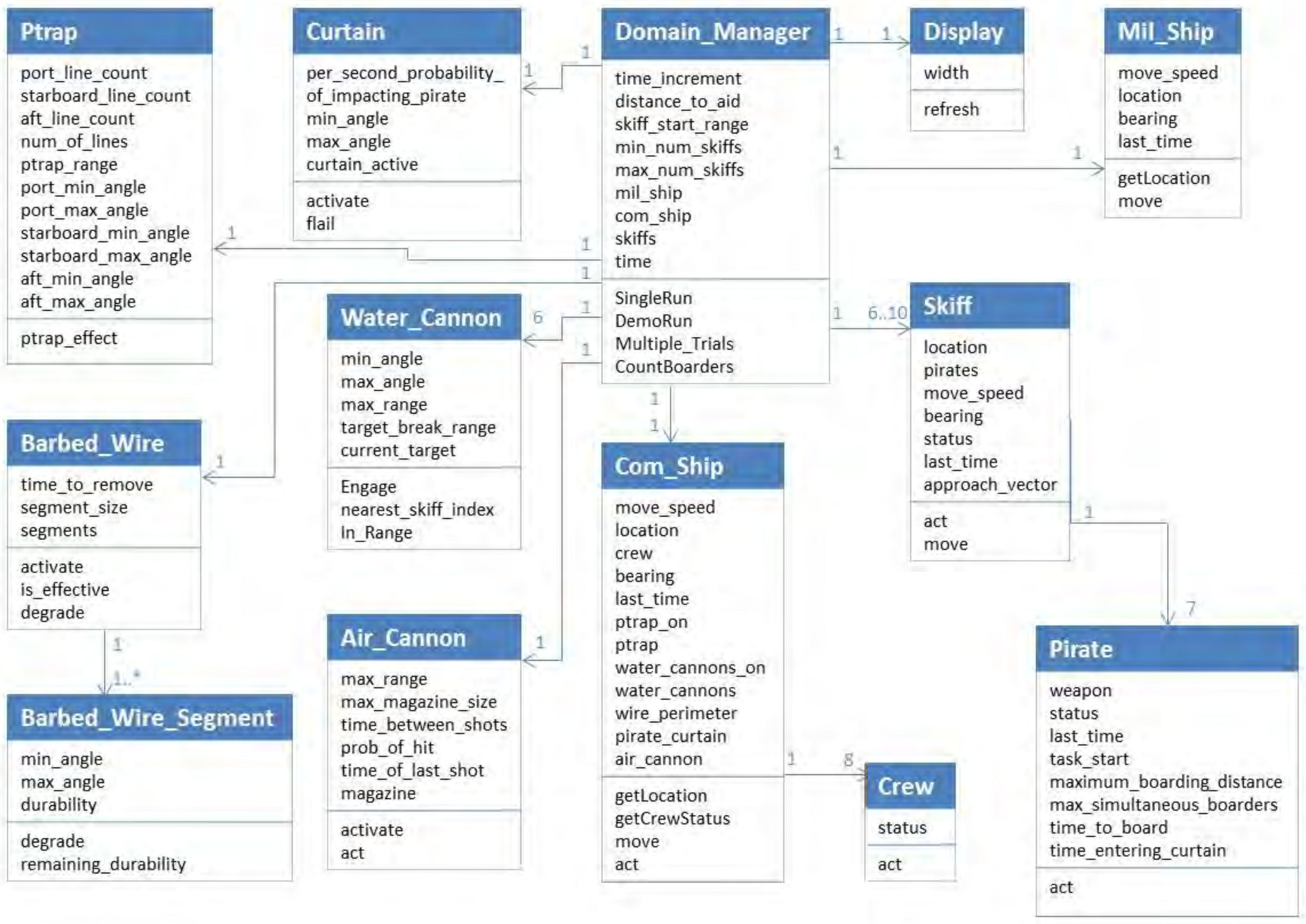
```
classdef Skiff_Status
    %Skiff_Status Enumeration of all possible states for the Skiff.
    %   Enumeration of all possible states for the Skiff.

    enumeration
        NORMAL
        DISABLED
        REPELLED
    end

end
```

Q. WATER_CANNON.M

```
classdef Water_Cannon
    %Water_Cannon Represents the water cannon countermeasure system within the model.
    %          the water cannon countermeasure features a remote-controlled water turret
that operates like a firehose in suppressing and forcing away pirates. The intended use
is to flood the skiffs, but it was determined that the pirates would seek to avoid this
eventuality so within the model the Water Cannon object acts to force skiffs out of its
range. The Water Cannons act on one skiff at a time and cause them to flee the Commercial
Ship's proximity. Each of the six Water Cannons is created as its own instance and tracks
its own tasking.

    properties
        min_angle; %the starting angle for the region of effect over which the Water
Cannon can act.
        max_angle; %the ending angle for the region of effect over which the Water Cannon
can act.
        max_range = 85; %the maximum distance from the Commercial Ship in meters at which
the Water Cannon can reach.
        target_break_range = 70; %the distance a Skiff must be pushed before the Water
Cannon will consider alternate targets.
        current_target; %the array index of the currently targeted Skiff.
    end

    methods
        function obj = Water_Cannon(min,max)
            %Constructor allocates memory, establishes pointers, and initialized object
members to starting values.

            obj.min_angle = min;
            obj.max_angle = max;
            obj.current_target = 0;
        end

        function [obj,skiffs] = Engage(obj,com_ship,skiffs)
            %the Water Cannon takes action; targeting a Skiff and pushing it back.

            %identify targets within range
            potential_targets = 0;
            for i = 1:size(skiffs,2)
                angle =  get_angle(com_ship.bearing,skiffs(i).location -
com_ship.location);
                if (angle > obj.min_angle) && (angle < obj.max_angle) &&
(norm(skiffs(i).location - com_ship.location) < obj.max_range) && (skiffs(i).status ~=
Skiff_Status.DISABLED)
                    if potential_targets
                        potential_targets(size(potential_targets,2)+1) = i;
                    else
                        potential_targets(1) = i;
                    end
                end
            end
            %if there is not currently a target
            if (~obj.current_target)
                %If there are potential targts select one as the current
                %target.
                if potential_targets
                    obj.current_target = obj.Get_Closest(com_ship.location, skiffs,
potential_targets);
                end

            elseif ~obj.In_Range(com_ship,skiffs(obj.current_target))
                % if the current target is
                %outside the maximum range then if there are potential targets
                %set the closest as the new target.
                %Free the current target if it has moved out of range

                skiffs(obj.current_target).status = Skiff_Status.NORMAL;
                obj.current_target = 0;


            else
                %cannon repels skiff provided ptrap has not just disabled
                %it
                if skiffs(obj.current_target).status ~= Skiff_Status.DISABLED
                    skiffs(obj.current_target).status = Skiff_Status.REPELLED;
                end

                %cannon knocks all boarders back
                for k = 1:size(skiffs(obj.current_target).pirates,2)
                    if skiffs(obj.current_target).pirates(k).status ==
Pirate_Status.ATTEMPTING_TO_BOARD
                        skiffs(obj.current_target).pirates(k).status =
Pirate_Status.IDLE;
                    end
                end
            end
        end

        function nearest_skiff_index = Get_Closest(obj, origin, skiffs,
allowable_indices)
            %Determines which skiff out of an array is the closest the selected origin
point.

            nearest_skiff_yet = 0;
            nearest_range_yet = 86;

            %Iterate through skiffs and keep track of the nearest one
            %checked yet.
            for j = 1:size(allowable_indices,2)
                if(norm(skiffs(allowable_indices(j)).location - origin) <
nearest_range_yet)
                    nearest_skiff_yet = allowable_indices(j);
                    nearest_range_yet = norm(skiffs(allowable_indices(j)).location -
origin);
                end
            end

            nearest_skiff_index = nearest_skiff_yet;
        end

        function in_range = In_Range(obj, com_ship, skiff)
            %Checks if a particular Skiff is within range of this Water Cannon both in
regards to distance and angular region.

            in_range = 0;
            angle =  get_angle(com_ship.bearing,skiff.location - com_ship.location);
            if (angle > obj.min_angle) && (angle < obj.max_angle) && (norm(skiff.location
- com_ship.location) < obj.max_range)
                in_range = 1;
            end
        end
    end

end
```

THIS PAGE INTENTIONALLY LEFT BLANK

APPENDIX C. MODEL UML CLASS RELATIONS



THIS PAGE INTENTIONALLY LEFT BLANK

APPENDIX D. MODEL ASSUMPTIONS

Assumption	Explanation	Rationale	Consequence of Incorrect Assumption	Reference
Pirate skiffs move directly at the commercial vessel, resulting in a chase pattern.	the two major movement patterns considered for the skiffs was the current aim-directly-at-the-ship or a more complex aim-where-the-ship-will-be. That is, it would be more efficient for the pirates to aim at a forward intercept point, but this requires they be able to accurately judge the ships and their own relative velocities in order to maintain the relative angle-off-bearing of the target.	the predictive behavior was considered challenging to accomplish on the seas from a small skiff and would have driven additional complexity into the model and so was deferred.	Would result in the proportion of pirate skiffs approaching from the rear of the ship in the model to non-representative.	None found
the commercial vessel begins the scenario 1,075 km from a ship that can provide effective aid (presumably a military vessel).		Jakarta and Belawan appear to be the two most distant naval bases from each other in the region of interest (about 2154 km). It is likely the pirates would try and attack ships at the worst place for them, which would be halfway between the two.	A dramatically shorter required distance to travel would result in time-delay countermeasures (such as razor wire) being more effective than currently represented within the model.	http://www.ordersofbattle.darkscape.net/site/maps/map_files/indonesia_navybases.gif

Assumption	Explanation	Rationale	Consequence of Incorrect Assumption	Reference
Pirate gun/rocket fire prior to boarding does little to degrade the commercial ship's capabilities.	With all countermeasures selected for the model either not needing active control or allowing for remote control from within the ship, it was presumed that the lack of crew exposure would render pirate weapons ineffective prior to boarding.	While the pirates have RPGs that could potentially cause structural damage to the ship; they are unlikely to fully disable or sink the ship as that would prevent them from profiting. The countermeasure system emplacements should present small enough a target that attacks against them are not a driving factor.	If the pirates are able to fire rockets from skiffs accurately enough to disable countermeasure emplacements then there is a significant dynamic to the system configurations that the model is not currently accounting for. Assuming accuracy is a factor of range the longer distance countermeasures would be preferred under those conditions.	
Pirate skiffs are first recognized as hostile at 1 mile.	the scenario begins with all pirate skiffs entering the recognition range simultaneously (considered a worst-case). This range is defined by how far out the ship can recognize them as pirates at and begin to respond to the attack.	the range is intended to represent the distance at which the pirates can be observed by binoculars to identify possibly armaments or other indicators of possible hostility.	Relatively little effect if the recognition distance is, in fact, greater than that used. If the distance is shorter however, some countermeasures may experience a smaller area over which they can be effective.	http://www.steiner-optics.com/binoculars/military/m50-lrf-military-10x50-lrf

Assumption	Explanation	Rationale	Consequence of Incorrect Assumption	Reference
the number of skiffs in an attack ranges from 6 to 10.	the model uses a uniform distribution of 6–10 pirate skiffs on each scenario.		Some countermeasures may be susceptible to breaking down if overwhelmed. In particular the P-trap could run out of lines and become entirely ineffective against additional skiffs.	
the pirate skiffs move at 12.86 meters per second		Based on a presumed maximum speed of 25 knots.	If pirate skiff speeds are in fact less than the commercial ship speeds then there would be significant changes in which countermeasures are preferred as the skiffs need only be fended off for one pass. If they are much faster than predicted then ranged countermeasures would have less time to be in effect before the skiff reached the hull.	

Assumption	Explanation	Rationale	Consequence of Incorrect Assumption	Reference
Skiffs carry 8 pirates each.	Each skiff contains 8 Pirate objects, one of which is immediately committed to piloting the skiff.		the only countermeasure currently in the model that is directly impacted by the number of pirates on board is the pirate curtain system. The results for that system are likely to change as a result of differing numbers of pirates per skiff.	
Razor wire and the pirate curtain are mutually supportive and non-interfering.	the model assumes that the razor/barbed wire can be placed such that the pirates remain within range of the pirate curtain while trying to bypass the wire, but also such that the pirate curtain never damages the wire.	there is insufficient data on the statistical movement patterns of the pirate curtain to understand the validity of this assumption. It is possible that there is not actually any placement where the wire would keep the pirates within the curtain's range without risking damage to the wire itself.	A synergy that the model represents between the razor wire and pirate curtain countermeasures would be reversed with the two in conflict and weakening each other instead of supporting.	More data is needed on the pirate curtain movement patterns.

Assumption	Explanation	Rationale	Consequence of Incorrect Assumption	Reference
the commercial vessel moves at 10.28 meters per second.		Based on a presumed maximum speed of 20 knots.	the relative speeds of the commercial vessel versus the skiffs affect the amount of time ranged countermeasures have to work on the skiffs as they approach. A significantly higher speed for the commercial ship may also allow the ship to pull away from the skiffs, or dramatically shorten the time needed to reach aid.	http://www.theguardian.com /environment/2010/jul/25/ slow-ships-cut-greenhouse –emissions
the military vessel moves at 15.55 meters per second.		Based on an advertised maximum speed of Arleigh-Burke class destroyer.	Slower speeds would increase the time the commercial ship must fend for itself; reducing the effectiveness of countermeasures that only delay without disabling.	http://www.navy.mil/navydata/fact_display.asp?cid=4200&tid=900&ct=4
Pirates can attempt to board the ship once their skiff is within 2 meters.	2 meters is the trigger chosen to start the pirate boarding actions.	Based on boarding attempts with ladders and grappling hooks. Presumed based on best-guess.	Little effect for changes within the same order of magnitude.	

Assumption	Explanation	Rationale	Consequence of Incorrect Assumption	Reference
Only 2 pirates from a single skiff may attempt to board at any given time.	It was believed that there would be some limitation on the amount of boarding gear (such as ladders) that could be fit within the skiff. Additionally, there are constraints to how many pirates can congregate against the same rail at any given time.	2 was a best assumption made with scarce data.	Would decrease the time barbed wire delays boarders (more people cutting at it) and would decrease the effectiveness of the pirate curtain.	
It takes an unhindered pirate 45 seconds to board the commercial vessel.	This constant accounts for the time needed for an unharrassed pirate to mount and climb a ladder or grappling line.	This is a strong assumption that was made in the absence of any identified study on the subject.	Would alter the effectiveness of the razor wire, pirate curtain and water cannon countermeasures at a minimum as well as changing how the baseline no-countermeasure scenario plays out.	
the air cannon can effectively fire 850 meters		Based on advertised capability.	Changes would directly impact the effectiveness of the air cannon.	http://www.bcbin.com/products/product_details.php?category=marine&product=SecurityBuccaneer#BUC001

Assumption	Explanation	Rationale	Consequence of Incorrect Assumption	Reference
the air cannon can fire six shots in any given attack.	Based on the presumption of a 6 shot clip with the crew being precluded from reloading it during the attack due to gunfire from the skiffs.	Assumed based on the size of the pictured device. A relatively strong assumption.	With only six shots the air cannon is easily overwhelmed by larger numbers of skiffs. If the true count is significantly higher in relation to the number of skiffs than the air cannon's effectiveness would greatly increase.	
the air cannon requires 10 seconds between shots.	This is largely based on a notional time needed to aim at a new target to some extent the physical workings of the air cannon may require time to rebuild air pressure or move the next projectile into position.	Actual data is not available. Arbitrarily assumed.	Relatively small changes if the true delay is within several seconds of the 10 currently used. Much larger delays would give pirate skiffs the opportunity to rush in between shots; lowering the cannon's effectiveness.	
the air cannon has a 0.5 probability of disabling a skiff with any given shot.	This metric takes into account the chance of the shot missing entirely and of the skiff managing to ignore the impediment.	No true data has been generated yet. 0.5 was selected to allow the model to take into account the miss chance.	Performance of the air cannon will scale directly with the probability per shot.	

Assumption	Explanation	Rationale	Consequence of Incorrect Assumption	Reference
It takes 120 seconds to remove or bypass the razor/barbed wire.	A two-minute delay period is associated with boarders disabling the wire. If multiple boarders are at the same segment they can work together to bypass it faster.	Two minutes was a best assumption. No study was found showing razor wire delay time directly. It is believed that the shape of the ship hull would preclude methods that are used to bypass wire on fences (i.e., throwing a rug over it).	Alters the effectiveness of system configurations where the wire is giving other countermeasures more time to work on the skiffs.	
the pirate curtain's probability of impacting an attempted boarder in its range is 0.05 each second.	Represents the chance each second of the flailing hose to impact a pirate on a ladder/rope in such a way as to disable or kill them.	Strong assumption. No studies are available and the dynamic motion of the hose is not described sufficiently to model in detail.	Directly affects the effectiveness of the curtain.	
the P-Trap can stop 10 skiffs in each of three regions: starboard, port and aft.	the P-Trap has 20 lines hanging off its starboard and port rails and another 20 aft of it. Each boat passing through the lines consumes 2–3 lines (worst case of 2). Thus, 20/2 = 10 skiffs stopped per side.	Based on a representative configuration selected from the supplier's advertised models.	Would scale the effectiveness of the P-trap.	https://www.google.com/patents/EP2459439B1?cl=en

Assumption	Explanation	Rationale	Consequence of Incorrect Assumption	Reference
P-Trap lines extend 10 meters from the hull of the ship.	Defines the region in with which the skiffs will be affected by the entangling lines. In reality the lines off the aft extend further (50-1000m) but treating the aft equivalently simplifies the model without much expected effect on the analysis.	the main driver for P-trap success is an adequate number of lines deployed. Defining the rear range as shorter than it is in reality in order to simplify the model will not change how many skiffs can be entangled.	None expected.	https://www.google.com/patents/EP2459439B1?cl=en
the water cannon's maximum range is 85 meters.		Based upon the advertised capability of a representative system.	May affect the effectiveness of the water cannon countermeasure.	http://www.unifire.com/sites/default/files/pliki/unifire_anti-pirate_water_cannon_system_brochure.pdf
the water cannon operator looks for a new target once a skiff has been forced out to 70m.	A defined distance at which the operator starts searching for a higher priority target to focus on. Prevents tunnel vision on one skiff while others approach unimpeded.	Likely that a tactical operator would try to prioritize nearby skiffs, but at the same time that some minimum distance must be reached for the current target for it to be safe to switch off them momentarily. Somewhat arbitrarily chosen based off those parameters.	Affects how the cannon performs when juggling between targets.	

THIS PAGE INTENTIONALLY LEFT BLANK

APPENDIX E. MODEL FUTURE WORK

- Refactor the model code for efficiency.
    - The Model was architected and programmed with a focus on simple, comprehensible logic. These attributes were prioritized in order to increase the likelihood of the model working correctly and being transferable to other users; but at the cost of prolonged run times. There are multiple examples of real-time memory allocation or calculation of intermediate values that could have been precalculated and hardcoded. Given the prolonged run times of the model in operation (around 1 hour for 100 runs), it would likely be worth making the code more efficient to increase usability.
- Utilize crew member tasking tracking.
    - An original intent of the model that fell out of scope in order to a) reduce complexity and runtime of the model and b) bring the modeling schedule back to the left, was to track what each crew member of the commercial ship was doing at any time. This would show limitations of user-controlled countermeasures with and without additional crew members. As the model exists today the task load on the ship's crew is not considered, to the perceived benefit of countermeasures that require active control.
- Test and refine implementation of the variable time increment feature of the model.
    - The model was originally designed to allow the user to select what time increment should be used in order to tradeoff between fidelity and run time. By default the model runs in one second increments. While this philosophy was maintained in the design and coding of the model, the feature was not included in the debugging and testing of the model due to schedule constraints. This helped bring the model schedule back in line with the project's, but has constrained users to the 1 second option.
- Incorporate pirate psychology/physical limitations.
    - The model currently assumes a mindless dedication by the pirates. They fight to the last man and for the entire trip to aid. It is likely that at some point prior to those conditions the pirates would abandon their attack. Whether it is running out of gas or losing too many skiffs, there are likely some conditions that would cause the pirates to "give up" that should be identified and incorporated.

LIST OF REFERENCES

"18 Anti-Piracy Weapons for Ships to Fight Pirates*."* 2013. *Marine Insight*. Retrieved 17, Aug 2014. http://www.marineinsight.com/marine/marine-news/headline/18-anti-piracy-weapons-for-ships-to-fight-pirates/.

Ben-Ari, Nirit. 2013. "Piracy In West Africa: A Bumpy Road to Maritime Security" *Daily News Egypt (Cairo)*, December 15.

Blanchard, Benjamin S., & Fabrycky, Wolter J. 2011. *Systems Engineering and Analysis.* Boston: Prentice Hall.

BMP4 Best Management Practices for Protection against Somalia Based Piracy. 2011. Edinburgh: Witherby Pub. Group. Retrieved January 12, 2014. http://eunavfor.eu/wpcontent/uploads/2013/01/bmp4-low-res_sept_5_20111.pdf.

Booton, Jennifer 2013. "Piracy Pays: Inside the Lucrative Fight to Foil High-Sea Hijackings." *FOX Business*, April 10.

Chalk, Peter. 2008. The *Maritime Dimension of International Security: Terrorism, Piracy, and Challenges For the United States.* Santa Monica: RAND Corporation.

CJCSI Instruction 3170.01.2007, Joint Capabilities Integration and Development System. 2007. Washington, D.C.: Department of Defense

Defense Acquisition Guidebook. 2013. Defense Acquition University. Retrieved Aug 17, 2014. https://dag.dau.mil/Pages/Default.aspx

Elleman, Bruce, Forbes andrew, & Rosenberg David. 2010. *Piracy and Maritime Crime: Historical and Modern Case Studies.* Newport Papers 35: 97–137.

"Enforcing the Law: An Economic Approach to Maritime Piracy and Its Control." 2014. *Hellenic Shipping News*. Retrieved July 23. http://www.hellenicshippingnews.com/14f77eeb-86e6-4252-bd03-92aeb0ab35fb/.

Hughey, Richard L. 2007. *Targeting At the Speed of Light.* Air War College. Retrieved Aug 17, 2014. http://www.au.af.mil/au/awc/awcgate/cst/bh_hughey.pdf.

Lilly, Trena C., & Russell, Bruce R. 2003. The *Multi-Mission Maritime Aircraft Design Reference Mission.* Johns Hopkins APL Technical Digest, 24(3): 257–262.

Lymer, David, Funge-Smith, Simon, & Greboval, Dominique. 2009. The *Fishing Fleet In Aceh Province, Indonesia.* Washington, D.C.: RAP Publication.

Madsen, Jens V., Seyle, Conor, Brandt, Kellie, Purser, Ben, Randall, Heather, & Roy, Kellie. 2014. *State of Maritime Policy 2013.* Colorado: OBP Oceans Beyond Piracy. Retrieved Aug 17, 2014. http://oceansbeyondpiracy.org/sites/default/files/attachments/SoP2013-Digital.pdf.

Manners, Andrew. 2014. *Pirates Release Ship, But Indonesia Still Worst For Piracy.* Future Directions International, June 4.

"Modern Piracy." 2014. Maritime Connector. Retrieved Aug 17. http://maritime-connector.com/wiki/piracy/.

Ong-Webb, Graham G. 2006. *Piracy, Maritime Terrorism and Securing the Malacca Straits.* Leiden, Netherlands: Institute of Southeast Asian Studies/International Institute for Asian Studies

Piracy and Armed Robbery Against Ships. 2014. ICC International Maritime Bureau. Retrieved Aug 17. www.kvnr.nl/stream/2013-annual-imb-piracy-report.

Ramones, Ikaika. 2013. "Pirates Take over the Waters In Indonesia." *Forbes Asia,* Aug 12.

Samatar, Ismail Abdi, Lindberg, Mark, & Mayhani, Basil. 2010. "The Dialectics of Piracy In Somalia: the Rich Versus the Poor." *Third World Quarterly*, 31(8), pp. 1377–1394.

The State of World Fisheries and Aquaculture. 2014. *FAO: Food and Agriculture Organization of the United Nations.* Retrieved Aug, 17, 2014. http://www.fao.org/3/a-i3720e.pdf.

Systems Engineering Process. 2014. Defense Acquition University. Retrieved May 22, 2014. https://dap.dau.mil/acquipedia/Pages/ArticleDetails.aspx?aid=9c591ad6-8f69-49dd-a61d-4096e7b3086c.

World Oil Transit Chokepoints. 2012. Energy Information Agency. http://www.eia.gov/countries/analysisbriefs/World_Oil_Transit_Chokepoints/wotc.pdf.

INITIAL DISTRIBUTION LIST

1. Defense Technical Information Center
   Ft. Belvoir, Virginia

2. Dudley Knox Library
   Naval Postgraduate School
   Monterey, California